SOURCE CODE of INTEL 8085 SIMULATOR (MICROSOFT VISUAL BASIC 2010)

c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormAssembler.Designer.vb

```
********************************************************************************
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormAssembler
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.components = New System.ComponentModel.Container()
        Dim resources As System.ComponentModel.ComponentResourceManager = New System.ComponentModel.ComponentResourceManager(GetType(FormAssembler))
        Me.rtbSrc = New System.Windows.Forms.RichTextBox()
        Me.MenuStrip1 = New System.Windows.Forms.MenuStrip()
        Me.FileToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.NewSourceFileToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.LoadSourceFileToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.SaveSourceFileToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.ExitToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.AssemblerToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.AssembleAndSaveHEXFileToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.AssembleAndLoadProgramToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.Timer1 = New System.Windows.Forms.Timer(Me.components)
        Me.MenuStrip1.SuspendLayout()
        Me.SuspendLayout()
        '
        'rtbSrc
        '
        Me.rtbSrc.BackColor = System.Drawing.Color.Black
        Me.rtbSrc.Dock = System.Windows.Forms.DockStyle.Fill
        Me.rtbSrc.Font = New System.Drawing.Font("Consolas", 14.25!, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.rtbSrc.ForeColor = System.Drawing.Color.Yellow
        Me.rtbSrc.Location = New System.Drawing.Point(0, 24)
        Me.rtbSrc.Name = "rtbSrc"
        Me.rtbSrc.Size = New System.Drawing.Size(784, 522)
        Me.rtbSrc.TabIndex = 0
        Me.rtbSrc.Text = ""
        Me.rtbSrc.WordWrap = False
        '
        'MenuStrip1
        '
        Me.MenuStrip1.Items.AddRange(New System.Windows.Forms.ToolStripItem() {Me.FileToolStripMenuItem, Me.AssemblerToolStripMenuItem})
        Me.MenuStrip1.Location = New System.Drawing.Point(0, 0)
        Me.MenuStrip1.Name = "MenuStrip1"
        Me.MenuStrip1.Size = New System.Drawing.Size(784, 24)
        Me.MenuStrip1.TabIndex = 1
        Me.MenuStrip1.Text = "MenuStrip1"
        '
```

```
        'FileToolStripMenuItem
        '
        Me.FileToolStripMenuItem.DropDownItems.AddRange(New System.Windows.Forms.ToolStripItem() {Me.
    NewSourceFileToolStripMenuItem, Me.LoadSourceFileToolStripMenuItem, Me.
    SaveSourceFileToolStripMenuItem, Me.ExitToolStripMenuItem})
        Me.FileToolStripMenuItem.Name = "FileToolStripMenuItem"
        Me.FileToolStripMenuItem.Size = New System.Drawing.Size(123, 20)
        Me.FileToolStripMenuItem.Text = "Assembler Source File"
        '
        'NewSourceFileToolStripMenuItem
        '
        Me.NewSourceFileToolStripMenuItem.Name = "NewSourceFileToolStripMenuItem"
        Me.NewSourceFileToolStripMenuItem.Size = New System.Drawing.Size(153, 22)
        Me.NewSourceFileToolStripMenuItem.Text = "New Source File"
        '
        'LoadSourceFileToolStripMenuItem
        '
        Me.LoadSourceFileToolStripMenuItem.Name = "LoadSourceFileToolStripMenuItem"
        Me.LoadSourceFileToolStripMenuItem.Size = New System.Drawing.Size(153, 22)
        Me.LoadSourceFileToolStripMenuItem.Text = "Load Source File"
        '
        'SaveSourceFileToolStripMenuItem
        '
        Me.SaveSourceFileToolStripMenuItem.Name = "SaveSourceFileToolStripMenuItem"
        Me.SaveSourceFileToolStripMenuItem.Size = New System.Drawing.Size(153, 22)
        Me.SaveSourceFileToolStripMenuItem.Text = "Save Source File"
        '
        'ExitToolStripMenuItem
        '
        Me.ExitToolStripMenuItem.Name = "ExitToolStripMenuItem"
        Me.ExitToolStripMenuItem.Size = New System.Drawing.Size(153, 22)
        Me.ExitToolStripMenuItem.Text = "Exit"
        '
        'AssemblerToolStripMenuItem
        '
        Me.AssemblerToolStripMenuItem.DropDownItems.AddRange(New System.Windows.Forms.ToolStripItem()
     {Me.AssembleAndSaveHEXFileToolStripMenuItem, Me.AssembleAndLoadProgramToolStripMenuItem})
        Me.AssemblerToolStripMenuItem.Name = "AssemblerToolStripMenuItem"
        Me.AssemblerToolStripMenuItem.Size = New System.Drawing.Size(68, 20)
        Me.AssemblerToolStripMenuItem.Text = "Assembler"
        '
        'AssembleAndSaveHEXFileToolStripMenuItem
        '
        Me.AssembleAndSaveHEXFileToolStripMenuItem.Name = "AssembleAndSaveHEXFileToolStripMenuItem"
        Me.AssembleAndSaveHEXFileToolStripMenuItem.Size = New System.Drawing.Size(209, 22)
        Me.AssembleAndSaveHEXFileToolStripMenuItem.Text = "Assemble and Save HEX file"
        '
        'AssembleAndLoadProgramToolStripMenuItem
        '
        Me.AssembleAndLoadProgramToolStripMenuItem.Name = "AssembleAndLoadProgramToolStripMenuItem"
        Me.AssembleAndLoadProgramToolStripMenuItem.Size = New System.Drawing.Size(209, 22)
        Me.AssembleAndLoadProgramToolStripMenuItem.Text = "Assemble and Load Program"
        '
        'Timer1
        '
        Me.Timer1.Enabled = True
        Me.Timer1.Interval = 500
        '
        'FormAssembler
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(784, 546)
        Me.Controls.Add(Me.rtbSrc)
        Me.Controls.Add(Me.MenuStrip1)
        Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
        Me.MainMenuStrip = Me.MenuStrip1
        Me.Name = "FormAssembler"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Assembler"
        Me.MenuStrip1.ResumeLayout(False)
```

```
        Me.MenuStrip1.PerformLayout()
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents rtbSrc As System.Windows.Forms.RichTextBox
    Friend WithEvents MenuStrip1 As System.Windows.Forms.MenuStrip
    Friend WithEvents FileToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents NewSourceFileToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents LoadSourceFileToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents SaveSourceFileToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents ExitToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents AssemblerToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents AssembleAndSaveHEXFileToolStripMenuItem As System.Windows.Forms.
    ToolStripMenuItem
    Friend WithEvents AssembleAndLoadProgramToolStripMenuItem As System.Windows.Forms.
    ToolStripMenuItem
    Friend WithEvents Timer1 As System.Windows.Forms.Timer
End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormAssembler.
    vb
********************************************************************************
Imports System.IO
Imports System.Text

Public Class FormAssembler
    Private fname As String = ""
    Private bDirty As Boolean = False

    Private Sub FormAssembler_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.
    FormClosingEventArgs) Handles Me.FormClosing
        If bDirty = True Then
            Dim dlgResult As DialogResult
            dlgResult = MessageBox.Show("The ASM file had been modified. Would you like to save it
    before exiting?", "File Save", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question)
            If dlgResult = DialogResult.Yes Then
                SaveASM()
            ElseIf dlgResult = DialogResult.Cancel Then
                e.Cancel = True
            End If
        End If
    End Sub
    Private Sub FormAssembler_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
        If Not (MdiParent Is Nothing) Then
            Me.MainMenuStrip.Hide()
        End If
        rtbSrc.Focus()
    End Sub

    Private Sub ExitToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles ExitToolStripMenuItem.Click
        Me.Close()
    End Sub

    Private Sub NewSourceFileToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System
    .EventArgs) Handles NewSourceFileToolStripMenuItem.Click
        ClearAsm()
    End Sub

    Private Sub ClearAsm()
        SaveASM()

        rtbSrc.Clear()
        rtbSrc.Focus()
        fname = ""
        bDirty = False
        RefreshTitle()
```

```
End Sub
Private Sub LoadSourceFileToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles LoadSourceFileToolStripMenuItem.Click
    LoadASM()
End Sub

Private Sub LoadASM()
    Try
        Dim frm As New OpenFileDialog()
        frm.Multiselect = False
        frm.Title = "Load Intel 8085 ASM Source file"
        frm.Filter = "ASM File (*.asm)|*.asm|All files (*.*)|*.*||"
        If frm.ShowDialog() = Windows.Forms.DialogResult.OK Then
            ClearAsm()

            fname = frm.FileName
            Dim sr As New StreamReader(fname)
            rtbSrc.Text = sr.ReadToEnd()
            sr.Close()
        End If
        RefreshTitle()
        bDirty = False
    Catch ex As Exception

    End Try
End Sub

Private Sub SaveASM()
    Try
        If bDirty = False Then
            Exit Sub
        End If

        If fname.Trim.Length < 1 Then
            Dim frm As New SaveFileDialog()
            frm.Title = "Save Intel 8085 ASM Source file"
            frm.Filter = "ASM File (*.asm)|*.asm|All files (*.*)|*.*||"
            If frm.ShowDialog() = DialogResult.OK Then
                fname = frm.FileName
            End If
        End If

        If fname.Trim.Length < 1 Then
            Exit Sub
        End If

        Dim sw As New StreamWriter(fname, False, System.Text.Encoding.ASCII)
        sw.Write(rtbSrc.Text)
        sw.Flush()
        sw.Close()
        bDirty = False
        RefreshTitle()
    Catch ex As Exception

    End Try
End Sub

Private Sub SaveSourceFileToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles SaveSourceFileToolStripMenuItem.Click
    SaveASM()
End Sub

Private Sub rtbSrc_KeyUp(ByVal sender As Object, ByVal e As System.Windows.Forms.KeyEventArgs)
Handles rtbSrc.KeyUp
    ProcessKey(e.KeyCode, e.Handled)
End Sub

Private Sub rtbSrc_TextChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles
rtbSrc.TextChanged
    bDirty = True
```

```
    End Sub

    Private Sub RefreshTitle()
        Try
            Dim strTitle As String
            strTitle = Me.Text

            If fname.Trim.Length < 1 Then
                If strTitle.IndexOf("[") = -1 Then

                Else
                    strTitle = strTitle.Substring(0, strTitle.IndexOf("["))

                End If
            Else
                If strTitle.IndexOf("[") = -1 Then
                Else
                    strTitle = strTitle.Substring(0, strTitle.IndexOf("["))
                End If
                strTitle += "[ " + fname + " ]"
            End If
            Me.Text = strTitle
        Catch ex As Exception

        End Try
    End Sub


    Private Sub ProcessKey(ByVal k As Keys, ByRef bDone As Boolean)
        If k = Keys.Tab Then
            Dim idx As Integer = rtbSrc.SelectionStart

            Dim str As String = rtbSrc.Text
            Dim str1 As String
            If idx <= 1 Then
                str1 = ""
            Else
                str1 = str.Substring(0, idx)
            End If

            Dim str2 As String
            If idx < str.Length Then
                str2 = str.Substring(idx)
            Else
                str2 = ""
            End If

            Dim strSpaces As String = ""
            Dim col As Integer
            col = idx - rtbSrc.GetFirstCharIndexOfCurrentLine()
            col = col Mod 5
            col = 5 - col
            For i As Integer = 1 To col
                strSpaces += " "
            Next
            rtbSrc.Text = String.Format("{0}{1}{2}", str1, strSpaces, str2)
            rtbSrc.SelectionStart = idx + strSpaces.Length
            bDone = True
        ElseIf k = Keys.F5 Then
            Me.AssembleAndSaveHEXFileToolStripMenuItem.PerformClick()
            bDone = True
        ElseIf k = Keys.F2 Then
            Me.LoadSourceFileToolStripMenuItem.PerformClick()
            bDone = True
        ElseIf k = Keys.F3 Then
            Me.SaveSourceFileToolStripMenuItem.PerformClick()
            bDone = True
        ElseIf k = Keys.F9 Then
            Me.AssembleAndLoadProgramToolStripMenuItem.PerformClick()
            bDone = True
        End If
```

```
    End Sub



    Private Sub AssembleAndSaveHEXFileToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles AssembleAndSaveHEXFileToolStripMenuItem.Click
        FormReport.Clear()
        Dim ll As New List(Of AsmLine)
        If AssembleProgram(ll) = True Then
            ' use hex encoder to save
            Try
                HexSave(ll)
                LoadMem(ll)
                WriteLine("Program saved and then successfully loaded to memory")
            Catch ex As Exception
                WriteLine(ex.Message)
            End Try
        End If

    End Sub

    Private Sub SaveHexDataToFile(ByVal f As StreamWriter, ByVal addr As Integer, ByVal data As List ↙
    (Of Integer))
        Dim dstr As String
        dstr = ""
        Dim startAddress As Integer
        startAddress = addr

        For Each dt As Integer In data
            dstr += String.Format("{0:X2}", dt)
            addr += 1

            If dstr.Length >= &H10 Then
                ' save the data
                SaveHexNow(f, startAddress, dstr)
                startAddress = addr
                dstr = ""
            End If
        Next
        If dstr.Length > 0 Then
            SaveHexNow(f, startAddress, dstr)
        End If
    End Sub
    Private Sub SaveHexNow(ByVal f As StreamWriter, ByVal sa As Integer, ByVal data As String)
        Dim str As String
        str = ":"
        str += String.Format("{0:X2}", data.Length)
        str += String.Format("{0:X4}", sa)
        str += String.Format("{0:X2}", 0)
        str += data.Trim
        str += String.Format("{0:X2}", CheckSum(str.Substring(1)))
        str = str.ToUpper().Trim
        f.WriteLine(str)

    End Sub
    Private Function CheckSum(ByVal str As String) As Integer
        Dim ss As String
        Dim ii As Integer
        Dim ck As Integer
        ck = 0
        For i As Integer = 0 To str.Length - 1 Step 2
            ii = 0
            Try
                ss = str.Substring(i, 2)
                ss = ss.Trim.ToUpper() + "H"

                ii = ParseInteger(ss)

            Catch ex As Exception
```

```
            End Try
            ck += ii
        Next
        ck = Not (ck)
        ck += 1
        ck = ck And &HFF
        Return ck
    End Function
    Private Sub HexSave(ByVal l As List(Of AsmLine))
        Dim sa As Integer
        Dim startAddress As Integer
        Dim data As New List(Of Integer)
        sa = -1
        startAddress = 0
        Dim f As New FileInfo(fname)
        Dim fx As New FileInfo(f.DirectoryName + "/" + f.Name.Replace(f.Extension, ".hex"))
        fx.Delete()
        Dim ff As New StreamWriter(fx.FullName, False, System.Text.Encoding.ASCII)
        For Each ll As AsmLine In l
            If sa = -1 Then
                sa = ll.addr
                startAddress = sa
                data.Clear()
            End If

            If ll.addr <> sa Then
                SaveHexDataToFile(ff, startAddress, data)
                data.Clear()
                sa = ll.addr
                startAddress = sa
            End If

            data.Add(ll.data)
            sa += 1
        Next
        If data.Count > 0 Then
            SaveHexDataToFile(ff, startAddress, data)
        End If
        ff.WriteLine(":00000001FF")
        ff.Flush()
        ff.Close()
    End Sub

    Private Sub LoadMem(ByVal l As List(Of AsmLine))
        Dim ms As MachineState = FormMain.GetMachineState
        For Each l1 As AsmLine In l
            ms.SetMemory(l1.addr, l1.data, True)
        Next
    End Sub

    Private Sub AssembleAndLoadProgramToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e ↙
    As System.EventArgs) Handles AssembleAndLoadProgramToolStripMenuItem.Click
        FormReport.Clear()

        Dim ll As New List(Of AsmLine)
        If AssembleProgram(ll) = True Then
            ' use loader to load
            LoadMem(ll)
        End If

    End Sub

    Structure AsmLine
        Public addr As Integer
        Public data As Integer
    End Structure

    Structure SrcLine
        Public addr As Integer
        Public st As String
    End Structure
```

```
    Structure AsmLabel
        Public addr As Integer
        Public lbl As String
    End Structure

    Private Sub WriteLine(ByVal str As String)
        FormReport.WriteLine(str)
    End Sub
    Private Function AssembleProgram(ByRef ll As List(Of AsmLine)) As Boolean
        Try
            WriteLine("Assembler stated...")
            WriteLine(String.Format("Program Name: {0}", fname))

            Dim strProgram As String
            strProgram = rtbSrc.Text
            strProgram = AsmPreprocess(strProgram)

            Dim llp As New List(Of String)
            TrimProg(strProgram, llp)

            Dim srcLines As New List(Of SrcLine)
            Dim jmpLabel As New List(Of AsmLabel)
            AsmPre2(llp, srcLines, jmpLabel)

            Asmble(srcLines, jmpLabel, ll)
            WriteLine("Program assembled successfully")
            Return True
        Catch ex As Exception
            WriteLine("Assembling failed.")
            WriteLine("Exception: " + ex.Message)
            Return False
        End Try

    End Function

    Private Sub PrepAsmByte(ByRef asm As List(Of AsmLine), ByVal addr As Integer, ByVal data1 As ↙
    Integer)
        Dim a As New AsmLine()
        a.addr = addr
        a.data = data1 And &HFF
        asm.Add(a)
    End Sub

    Private Sub PrepAsmInt(ByRef asm As List(Of AsmLine), ByVal addr As Integer, ByVal data2 As ↙
    Integer)
        Dim l, h As Integer
        l = data2 And &HFF
        h = (data2 >> 8) And &HFF
        PrepAsmByte(asm, addr, l)
        PrepAsmByte(asm, addr + 1, h)
    End Sub

    Private Sub PrepAsm1(ByRef asm As List(Of AsmLine), ByVal addr As Integer, ByVal ins As Integer)
        PrepAsmByte(asm, addr, ins)
    End Sub

    Private Sub PrepAsm2(ByRef asm As List(Of AsmLine), ByVal addr As Integer, ByVal ins As Integer, ↙
    ByVal databyte As Integer)
        PrepAsmByte(asm, addr, ins)
        PrepAsmByte(asm, addr + 1, databyte)
    End Sub

    Private Sub PrepAsm3(ByRef asm As List(Of AsmLine), ByVal addr As Integer, ByVal ins As Integer, ↙
    ByVal dataint As Integer)
        PrepAsmByte(asm, addr, ins)
        PrepAsmInt(asm, addr + 1, dataint)
    End Sub

    Private Function IsAsm(ByVal s As String, ByRef sData As String, ByVal ins As String) As Boolean
        s = s.Trim()
```

```
        Dim i As Integer
        i = s.ToUpper().IndexOf(ins.ToUpper())
        If i = -1 Then
            Return False
        ElseIf i = 0 Then
            ' found it
        Else
            Return False
        End If
        s = s.Remove(0, ins.Length)
        If s.Length = 0 Then
            ' found it
        Else
            If s.Chars(0) = " " Then
                'found it
            Else
                Return False
            End If
        End If

        s = s.Trim
        sData = s
        Return True
    End Function

    Private Function DetReg(ByVal r As String) As Integer
        r = r.Trim.ToUpper
        Dim s() As String = {"B", "C", "D", "E", "H", "L", "M", "A"}
        For i As Integer = 0 To s.Length - 1
            If s(i).IndexOf(r) = 0 Then
                Return i
            End If
        Next
        Return -1
    End Function
    Private Function DetRegp(ByVal r As String) As Integer
        r = r.Trim.ToUpper
        Dim s() As String = {"B", "D", "H", "SP"}
        For i As Integer = 0 To s.Length - 1
            If s(i).IndexOf(r) = 0 Then
                Return i
            End If
        Next
        Return -1
    End Function
    Private Function DetRegp_psw(ByVal r As String) As Integer
        r = r.Trim.ToUpper
        Dim s() As String = {"B", "D", "H", "PSW"}
        For i As Integer = 0 To s.Length - 1
            If s(i).IndexOf(r) = 0 Then
                Return i
            End If
        Next
        Return -1
    End Function

    Private Sub EmptyStrException(ByVal str As String, ByVal src As SrcLine)
        If str.Trim.Length < 1 Then
            Exit Sub
        End If

        Throw New Exception(String.Format("Wrong syntax of instruction specified. {0:X4} {1}", src.  ↙
    addr, src.st))
    End Sub

    Private Function JGetAddress(ByVal str As String, ByVal lbl As List(Of AsmLabel)) As Integer
        Try
            For Each lb As AsmLabel In lbl
                If lb.lbl.Trim.ToUpper = str.ToUpper Then
                    Return lb.addr
                End If
```

```
            Next

            Return ParseInteger(str.Trim)
        Catch ex As Exception
            Throw New Exception("Cannot find the Jump label")
        End Try
    End Function
    Private Sub Asmble(ByVal src As List(Of SrcLine), ByVal lbl As List(Of AsmLabel), ByRef asm As
    List(Of AsmLine))
        WriteLine("Code Generation started...")
        For Each s As SrcLine In src
            Dim str As String = ""
            Dim sr, ds As String
            Dim csr, cds, c, data As Integer
            If IsAsm(s.st, str, "DB") = True Then
                PrepAsmByte(asm, s.addr, ParseInteger(str))
            ElseIf IsAsm(s.st, str, "MOV") = True Then
                ds = str.Substring(0, str.IndexOf(",")).Trim
                sr = str.Substring(str.IndexOf(",") + 1).Trim
                csr = DetReg(sr.ToUpper())
                cds = DetReg(ds.ToUpper())
                If (csr = -1) Or (cds = -1) Then
                    Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
    st))
                End If
                c = csr Or (cds << 3) Or (1 << 6)
                PrepAsm1(asm, s.addr, c)
            ElseIf IsAsm(s.st, str, "XCHG") = True Then
                EmptyStrException(str, s)
                PrepAsm1(asm, s.addr, &HEB)
            ElseIf IsAsm(s.st, str, "MVI") = True Then
                ds = str.Substring(0, str.IndexOf(",")).Trim.ToUpper()
                sr = str.Substring(str.IndexOf(",") + 1).Trim
                data = JGetAddress(sr, lbl)
                cds = DetReg(ds)
                If (cds = -1) Then
                    Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
    st))
                End If
                c = &H6 Or (cds << 3)
                PrepAsm2(asm, s.addr, c, data)
            ElseIf IsAsm(s.st, str, "LXI") = True Then
                ds = str.Substring(0, str.IndexOf(",")).Trim.ToUpper()
                sr = str.Substring(str.IndexOf(",") + 1).Trim
                data = JGetAddress(sr, lbl)
                cds = DetRegp(ds)
                If (cds = -1) Then
                    Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
    st))
                End If

                c = &H1 Or (cds << 4)
                PrepAsm3(asm, s.addr, c, data)
            ElseIf IsAsm(s.st, str, "LDAX") = True Then
                ds = str.Trim.ToUpper()
                cds = DetRegp(ds)
                If (cds = -1) Then
                    Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
    st))
                End If
                c = &HA Or (cds << 4)
                PrepAsm1(asm, s.addr, c)
            ElseIf IsAsm(s.st, str, "STAX") = True Then
                ds = str.Trim.ToUpper()
                cds = DetRegp(ds)
                If (cds = -1) Then
                    Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
    st))
                End If
                c = &H2 Or (cds << 4)
                PrepAsm1(asm, s.addr, c)
```

```
            ElseIf IsAsm(s.st, str, "LHLD") = True Then
                sr = str.Trim.ToUpper()
                data = JGetAddress(sr, lbl)
                PrepAsm3(asm, s.addr, &H2A, data)
            ElseIf IsAsm(s.st, str, "SHLD") = True Then
                sr = str.Trim.ToUpper()
                data = JGetAddress(sr, lbl)
                PrepAsm3(asm, s.addr, &H22, data)
            ElseIf IsAsm(s.st, str, "LDA") = True Then
                sr = str.Trim.ToUpper()
                data = JGetAddress(sr, lbl)
                PrepAsm3(asm, s.addr, &H3A, data)
            ElseIf IsAsm(s.st, str, "STA") = True Then
                sr = str.Trim.ToUpper()
                data = JGetAddress(sr, lbl)
                PrepAsm3(asm, s.addr, &H32, data)
            ElseIf IsAsm(s.st, str, "ADD") = True Then
                sr = str.Trim.ToUpper()
                csr = DetReg(sr)
                If (csr = -1) Then
                    Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
    st))
                End If
                c = &H80 Or csr
                PrepAsm1(asm, s.addr, c)
            ElseIf IsAsm(s.st, str, "ADC") = True Then
                sr = str.Trim.ToUpper()
                csr = DetReg(sr)
                If (csr = -1) Then
                    Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
    st))
                End If
                c = &H88 Or csr
                PrepAsm1(asm, s.addr, c)
            ElseIf IsAsm(s.st, str, "SUB") = True Then
                sr = str.Trim.ToUpper()
                csr = DetReg(sr)
                If (csr = -1) Then
                    Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
    st))
                End If
                c = &H90 Or csr
                PrepAsm1(asm, s.addr, c)
            ElseIf IsAsm(s.st, str, "SBB") = True Then
                sr = str.Trim.ToUpper()
                csr = DetReg(sr)
                If (csr = -1) Then
                    Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
    st))
                End If
                c = &H98 Or csr
                PrepAsm1(asm, s.addr, c)
            ElseIf IsAsm(s.st, str, "INR") = True Then
                sr = str.Trim.ToUpper()
                csr = DetReg(sr)
                If (csr = -1) Then
                    Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
    st))
                End If
                c = &H4 Or (csr << 3)
                PrepAsm1(asm, s.addr, c)
            ElseIf IsAsm(s.st, str, "DCR") = True Then
                sr = str.Trim.ToUpper()
                csr = DetReg(sr)
                If (csr = -1) Then
                    Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
    st))
                End If
                c = &H5 Or (csr << 3)
                PrepAsm1(asm, s.addr, c)
            ElseIf IsAsm(s.st, str, "ANA") = True Then
```

```
            sr = str.Trim.ToUpper()
            csr = DetReg(sr)
            If (csr = -1) Then
                Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
st))
            End If
            c = &HA0 Or csr
            PrepAsm1(asm, s.addr, c)
        ElseIf IsAsm(s.st, str, "XRA") = True Then
            sr = str.Trim.ToUpper()
            csr = DetReg(sr)
            If (csr = -1) Then
                Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
st))
            End If
            c = &HA8 Or csr
            PrepAsm1(asm, s.addr, c)
        ElseIf IsAsm(s.st, str, "ORA") = True Then
            sr = str.Trim.ToUpper()
            csr = DetReg(sr)
            If (csr = -1) Then
                Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
st))
            End If
            c = &HB0 Or csr
            PrepAsm1(asm, s.addr, c)
        ElseIf IsAsm(s.st, str, "CMP") = True Then
            sr = str.Trim.ToUpper()
            csr = DetReg(sr)
            If (csr = -1) Then
                Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
st))
            End If
            c = &HB8 Or csr
            PrepAsm1(asm, s.addr, c)
        ElseIf IsAsm(s.st, str, "DAD") = True Then
            sr = str.Trim.ToUpper()
            csr = DetRegp(sr)
            If (csr = -1) Then
                Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
st))
            End If
            c = &H9 Or (csr << 4)
            PrepAsm1(asm, s.addr, c)
        ElseIf IsAsm(s.st, str, "INX") = True Then
            sr = str.Trim.ToUpper()
            csr = DetRegp(sr)
            If (csr = -1) Then
                Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
st))
            End If
            c = &H3 Or (csr << 4)
            PrepAsm1(asm, s.addr, c)
        ElseIf IsAsm(s.st, str, "DCX") = True Then
            sr = str.Trim.ToUpper()
            csr = DetRegp(sr)
            If (csr = -1) Then
                Throw New Exception(String.Format("Invalid operand at {0:X4} : {1}", s.addr, s.
st))
            End If
            c = &HB Or (csr << 4)
            PrepAsm1(asm, s.addr, c)
        ElseIf IsAsm(s.st, str, "DAA") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &H27)
        ElseIf IsAsm(s.st, str, "CMA") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &H2F)
        ElseIf IsAsm(s.st, str, "STC") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &H37)
```

```
            ElseIf IsAsm(s.st, str, "CMC") = True Then
                EmptyStrException(str, s)
                PrepAsm1(asm, s.addr, &H3F)
            ElseIf IsAsm(s.st, str, "RLC") = True Then
                EmptyStrException(str, s)
                PrepAsm1(asm, s.addr, &H7)
            ElseIf IsAsm(s.st, str, "RRC") = True Then
                EmptyStrException(str, s)
                PrepAsm1(asm, s.addr, &HF)
            ElseIf IsAsm(s.st, str, "RAL") = True Then
                EmptyStrException(str, s)
                PrepAsm1(asm, s.addr, &H17)
            ElseIf IsAsm(s.st, str, "RAR") = True Then
                EmptyStrException(str, s)
                PrepAsm1(asm, s.addr, &H1F)
            ElseIf IsAsm(s.st, str, "ADI") = True Then
                data = JGetAddress(str.Trim(), lbl)
                PrepAsm2(asm, s.addr, &HC6, data)
            ElseIf IsAsm(s.st, str, "ACI") = True Then
                data = JGetAddress(str.Trim(), lbl)
                PrepAsm2(asm, s.addr, &HCE, data)
            ElseIf IsAsm(s.st, str, "SUI") = True Then
                data = JGetAddress(str.Trim(), lbl)
                PrepAsm2(asm, s.addr, &HD6, data)
            ElseIf IsAsm(s.st, str, "SBI") = True Then
                data = JGetAddress(str.Trim(), lbl)
                PrepAsm2(asm, s.addr, &HDE, data)
            ElseIf IsAsm(s.st, str, "ANI") = True Then
                data = JGetAddress(str.Trim(), lbl)
                PrepAsm2(asm, s.addr, &HE6, data)
            ElseIf IsAsm(s.st, str, "XRI") = True Then
                data = JGetAddress(str.Trim(), lbl)
                PrepAsm2(asm, s.addr, &HEE, data)
            ElseIf IsAsm(s.st, str, "ORI") = True Then
                data = JGetAddress(str.Trim(), lbl)
                PrepAsm2(asm, s.addr, &HF6, data)
            ElseIf IsAsm(s.st, str, "CPI") = True Then
                data = JGetAddress(str.Trim(), lbl)
                PrepAsm2(asm, s.addr, &HFE, data)
            ElseIf IsAsm(s.st, str, "JMP") = True Then
                data = JGetAddress(str, lbl)
                PrepAsm3(asm, s.addr, &HC3, data)
            ElseIf IsAsm(s.st, str, "JNZ") = True Then
                data = JGetAddress(str, lbl)
                PrepAsm3(asm, s.addr, &HC2, data)
            ElseIf IsAsm(s.st, str, "JZ") = True Then
                data = JGetAddress(str, lbl)
                PrepAsm3(asm, s.addr, &HCA, data)
            ElseIf IsAsm(s.st, str, "JNC") = True Then
                data = JGetAddress(str, lbl)
                PrepAsm3(asm, s.addr, &HD2, data)
            ElseIf IsAsm(s.st, str, "JC") = True Then
                data = JGetAddress(str, lbl)
                PrepAsm3(asm, s.addr, &HDA, data)
            ElseIf IsAsm(s.st, str, "JPO") = True Then
                data = JGetAddress(str, lbl)
                PrepAsm3(asm, s.addr, &HE2, data)
            ElseIf IsAsm(s.st, str, "JPE") = True Then
                data = JGetAddress(str, lbl)
                PrepAsm3(asm, s.addr, &HEA, data)
            ElseIf IsAsm(s.st, str, "JP") = True Then
                data = JGetAddress(str, lbl)
                PrepAsm3(asm, s.addr, &HF2, data)
            ElseIf IsAsm(s.st, str, "JM") = True Then
                data = JGetAddress(str, lbl)
                PrepAsm3(asm, s.addr, &HFA, data)
            ElseIf IsAsm(s.st, str, "PCHL") = True Then
                EmptyStrException(str, s)
                PrepAsm1(asm, s.addr, &HE9)
            ElseIf IsAsm(s.st, str, "CALL") = True Then
                data = JGetAddress(str, lbl)
```

```
            PrepAsm3(asm, s.addr, &HCD, data)
        ElseIf IsAsm(s.st, str, "CNZ") = True Then
            data = JGetAddress(str, lbl)
            PrepAsm3(asm, s.addr, &HC4, data)
        ElseIf IsAsm(s.st, str, "CZ") = True Then
            data = JGetAddress(str, lbl)
            PrepAsm3(asm, s.addr, &HCC, data)
        ElseIf IsAsm(s.st, str, "CNC") = True Then
            data = JGetAddress(str, lbl)
            PrepAsm3(asm, s.addr, &HD4, data)
        ElseIf IsAsm(s.st, str, "CC") = True Then
            data = JGetAddress(str, lbl)
            PrepAsm3(asm, s.addr, &HDC, data)
        ElseIf IsAsm(s.st, str, "CPO") = True Then
            data = JGetAddress(str, lbl)
            PrepAsm3(asm, s.addr, &HE4, data)
        ElseIf IsAsm(s.st, str, "CPE") = True Then
            data = JGetAddress(str, lbl)
            PrepAsm3(asm, s.addr, &HEC, data)
        ElseIf IsAsm(s.st, str, "CP") = True Then
            data = JGetAddress(str, lbl)
            PrepAsm3(asm, s.addr, &HF4, data)
        ElseIf IsAsm(s.st, str, "CM") = True Then
            data = JGetAddress(str, lbl)
            PrepAsm3(asm, s.addr, &HFC, data)
        ElseIf IsAsm(s.st, str, "RET") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &HC9)
        ElseIf IsAsm(s.st, str, "RNZ") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &HC0)
        ElseIf IsAsm(s.st, str, "RZ") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &HC8)
        ElseIf IsAsm(s.st, str, "RNC") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &HD0)
        ElseIf IsAsm(s.st, str, "RC") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &HD8)
        ElseIf IsAsm(s.st, str, "RPO") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &HD0)
        ElseIf IsAsm(s.st, str, "RPE") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &HE8)
        ElseIf IsAsm(s.st, str, "RP") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &HF0)
        ElseIf IsAsm(s.st, str, "RM") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &HF8)
        ElseIf IsAsm(s.st, str, "RST") = True Then
            ds = str
            cds = Integer.Parse(ds)
            If (cds >= 0) And (cds <= 7) Then
                Throw New Exception(String.Format("RST 0-7 is the only possible input for {0:X4}: ↙
 {1}", s.addr, s.st))
            End If
            c = &HC7 Or ((cds And &H7) << 4)
            PrepAsm1(asm, s.addr, c)
        ElseIf IsAsm(s.st, str, "PUSH") = True Then
            cds = DetRegp_psw(str)
            If cds = -1 Then
                Throw New Exception(String.Format("Register specified is invalid. {0:X4}:{1}", s. ↙
addr, s.st))
            End If
            c = &HC5 Or ((cds And &H3) << 4)
            PrepAsm1(asm, s.addr, c)
        ElseIf IsAsm(s.st, str, "POP") = True Then
            cds = DetRegp_psw(str)
```

```
            If cds = -1 Then
                Throw New Exception(String.Format("Register specified is invalid. {0:X4}:{1}", s.↙
addr, s.st))
            End If
            c = &HC1 Or ((cds And &H3) << 4)
            PrepAsm1(asm, s.addr, c)
        ElseIf IsAsm(s.st, str, "XTHL") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &HE3)
        ElseIf IsAsm(s.st, str, "SPHL") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &HF9)
        ElseIf IsAsm(s.st, str, "OUT") = True Then
            data = ParseInteger(str)
            PrepAsm2(asm, s.addr, &HD3, data)
        ElseIf IsAsm(s.st, str, "IN") = True Then
            data = ParseInteger(str)
            PrepAsm2(asm, s.addr, &HDB, data)
        ElseIf IsAsm(s.st, str, "DI") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &HF3)
        ElseIf IsAsm(s.st, str, "EI") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &HFB)
        ElseIf IsAsm(s.st, str, "NOP") = True Then
            PrepAsm1(asm, s.addr, &H0)
            EmptyStrException(str, s)
        ElseIf IsAsm(s.st, str, "HLT") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &H76)
        ElseIf IsAsm(s.st, str, "RIM") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &H20)
        ElseIf IsAsm(s.st, str, "SIM") = True Then
            EmptyStrException(str, s)
            PrepAsm1(asm, s.addr, &H30)
        Else
            Throw New Exception(String.Format("Unknown mnemonics encountered at {0:X4}H : {1}", s↙
.addr, s.st))
        End If

    Next

    WriteLine("Code generation completed successfully...")
    WriteLine(String.Format("Code size: {0} bytes", asm.Count))
End Sub

Private Function IsJumpLabelDetected(ByVal cmd As String, ByRef restOfString As String, ByRef    ↙
jLab As String) As Boolean
    cmd = cmd.Trim()
    Dim idx As Integer
    idx = cmd.IndexOf(":")
    If idx = -1 Then
        restOfString = cmd
        Return False
    End If

    Dim s1 As String
    s1 = cmd.Substring(0, idx)

    Dim s1c As Char()
    s1c = s1.ToCharArray()
    ' single quote, double quote, sp chars, any symbol is not allowed
    Dim bj As Boolean = True
    For Each s1ce As Char In s1c
        If Asc(s1ce) < 36 Then
            bj = False
            Exit For
        ElseIf (Asc(s1ce) > 36) And (Asc(s1ce) < 48) Then
            bj = False
            Exit For
```

```
            ElseIf (Asc(s1ce) > 57) And (Asc(s1ce) < 64) Then
                bj = False
                Exit For
            ElseIf (Asc(s1ce) > 90) And (Asc(s1ce) < 95) Then
                bj = False
                Exit For
            ElseIf (Asc(s1ce) > 95) And (Asc(s1ce) < 97) Then
                bj = False
                Exit For
            ElseIf (Asc(s1ce) > 122) Then
                bj = False
                Exit For
            End If
        Next
        If (Asc(s1c(0)) >= 48) And (Asc(s1c(0)) <= 57) Then
            bj = False
        End If

        If bj = False Then
            restOfString = cmd
            Return False
        End If

        jLab = cmd.Substring(0, idx)
        jLab = jLab.Trim

        cmd = cmd.Remove(0, idx)
        If cmd.Length <= 1 Then
            restOfString = ""
        Else
            restOfString = cmd.Substring(1)
        End If
        restOfString = restOfString.Trim
        Return True
    End Function
    Private Sub AsmPre2(ByVal ll As List(Of String), ByRef ln As List(Of SrcLine), ByRef jmp As List ↙
    (Of AsmLabel))
        WriteLine("Preprocessing stage 2 started...")
        ' process and remove org, .org, convert db, dw, ds to db 1 line each, removing everythine    ↙
    else, detect and remove jump label
        Dim sa As Integer = 0
        WriteLine(String.Format("Start Address: {0:X4}", sa))
        Dim il As Integer = 0
        For Each Str As String In ll
            il += 1
            Dim rcmd As String = ""
            Dim rjlb As String = ""
            Dim rpm As String = ""

            If IsJumpLabelDetected(Str, rcmd, rjlb) = True Then
                Dim al As New AsmLabel()
                al.lbl = rjlb.ToUpper().Trim()
                al.addr = sa
                jmp.Add(al)
            End If
            Try
                If IsAsm(rcmd, rpm, "DB") = True Then
                    While rpm.Length > 0
                        Dim ii As Integer
                        ii = rpm.IndexOf(",")
                        Dim ival As String
                        ival = ""
                        If ii = -1 Then
                            ival = rpm
                            rpm = ""
                        Else
                            ival = rpm.Substring(0, ii)
                            rpm = rpm.Remove(0, ii + 1)
                        End If
                        ival = ival.Trim()
```

```
                    Dim iival As Integer
                    iival = ParseInteger(ival)

                    Dim ss As New SrcLine()
                    ss.addr = sa
                    ss.st = String.Format("DB {0:X2}H", iival)
                    ln.Add(ss)
                    sa += 1
                End While
            ElseIf IsAsm(rcmd, rpm, "DW") = True Then
                While rpm.Length > 0
                    Dim ii As Integer
                    ii = rpm.IndexOf(",")
                    Dim ival As String
                    ival = ""
                    If ii = -1 Then
                        ival = rpm
                        rpm = ""
                    Else
                        ival = rpm.Substring(0, ii)
                        rpm = rpm.Remove(0, ii + 1)
                    End If
                    ival = ival.Trim()

                    Dim iival As Integer
                    iival = ParseInteger(ival)

                    Dim ss As New SrcLine()
                    ss.addr = sa
                    ss.st = String.Format("DB {0:X2}H", iival And &HFF)
                    ln.Add(ss)
                    sa += 1
                    ss = New SrcLine()
                    ss.addr = sa
                    ss.st = String.Format("DB {0:X2}H", (iival >> 8) And &HFF)
                    ln.Add(ss)
                    sa += 1
                End While
            ElseIf IsAsm(rcmd, rpm, "DS") = True Then
                Dim iid1, iid2 As Integer
                iid1 = rpm.IndexOf(Chr(34))
                iid2 = rpm.LastIndexOf(Chr(34))
                If (iid1 = -1) Or (iid2 = -1) Or (iid2 <= iid1) Then
                    Throw New Exception("Character string is not properly enclosed within ↙
quotation marks")
                End If
                Dim iisd As String
                iisd = rpm.Substring(iid1 + 1, iid2 - iid1 - 1)
                For Each iisdc As Char In iisd.ToCharArray()
                    Dim ss As New SrcLine()
                    ss.addr = sa
                    ss.st = String.Format("DB {0:X2}H", Asc(iisdc))
                    ln.Add(ss)
                    sa += 1
                Next
            ElseIf IsAsm(rcmd, rpm, "EQU") = True Then
                Dim iaddr As Integer
                iaddr = ParseInteger(rpm)

                Dim rt1, rt2 As String
                rt1 = ""
                rt2 = ""
                If IsJumpLabelDetected(Str, rt1, rt2) = False Then
                    Throw New Exception("EQU is used to assign predetermined address to a label. ↙
A label name is not found")
                End If

                Dim jt As AsmLabel
                jt.addr = iaddr
                jt.lbl = rt2.ToUpper
                For iaddr = 0 To (jmp.Count - 1)
```

```
                    If jmp(iaddr).lbl = jt.lbl Then
                        jmp.RemoveAt(iaddr)
                        Exit For
                    End If
                Next
                jmp.Add(jt)
            ElseIf IsAsm(rcmd, rpm, "ORG") = True Then
                Dim iaddr As Integer
                iaddr = ParseInteger(rpm)
                sa = iaddr

                Dim rt1, rt2 As String
                rt1 = ""
                rt2 = ""
                If IsJumpLabelDetected(Str, rt1, rt2) = True Then
                    Dim jt As AsmLabel
                    jt.addr = iaddr
                    jt.lbl = rt2.ToUpper
                    For iaddr = 0 To (jmp.Count - 1)
                        If jmp(iaddr).lbl = jt.lbl Then
                            jmp.RemoveAt(iaddr)
                            Exit For
                        End If
                    Next
                    jmp.Add(jt)
                End If

            Else
                Dim ilen As Integer
                ilen = GenerateAddress(rcmd)
                Dim sl As New SrcLine()
                sl.addr = sa
                sl.st = rcmd.Trim.ToUpper
                ln.Add(sl)

                sa += ilen
                End If
        Catch ex As Exception
            Throw New Exception(String.Format("ERROR IN LINE # {0} : {1}   {2}", il, Str, ex.     ↙
Message))
        End Try

    Next
    WriteLine(String.Format("End Address: {0:X4}", sa))

    WriteLine("Preprocessing stage 2 ended successfully.")
End Sub


Private Function ParseInteger(ByVal str As String) As Integer
    str = str.Trim.ToUpper
    If str.Length < 1 Then
        Throw New Exception("Blank string passed to ParseInteger(...)")
    End If
    If str.IndexOf("H") = -1 Then
        Return Integer.Parse(str)
    Else
        Return Integer.Parse(str.Substring(0, str.IndexOf("H")), Globalization.NumberStyles.      ↙
HexNumber)
    End If
End Function

Private Function GenerateAddress(ByVal str As String) As Integer
    str = str.Trim
    If str.Length < 1 Then
        Return 0
    End If

    str = str.ToUpper()
    Dim ostr As String
    ostr = ""
```

```
        If IsAsm(str, ostr, "MOV") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "XCHG") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "MVI") = True Then
            Return 2
        ElseIf IsAsm(str, ostr, "LXI") = True Then
            Return 3
        ElseIf IsAsm(str, ostr, "LDAX") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "LHLD") = True Then
            Return 3
        ElseIf IsAsm(str, ostr, "LDA") = True Then
            Return 3
        ElseIf IsAsm(str, ostr, "STAX") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "SHLD") = True Then
            Return 3
        ElseIf IsAsm(str, ostr, "STA") = True Then
            Return 3
        ElseIf IsAsm(str, ostr, "ADD") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "ADC") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "SUB") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "SBB") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "DAD") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "INR") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "INX") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "DCR") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "DCX") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "DAA") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "CMA") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "STC") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "CMC") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "RLC") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "RRC") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "RAL") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "RAR") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "ANA") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "XRA") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "ORA") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "CMP") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "ADI") = True Then
            Return 2
        ElseIf IsAsm(str, ostr, "ACI") = True Then
            Return 2
        ElseIf IsAsm(str, ostr, "SUI") = True Then
            Return 2
        ElseIf IsAsm(str, ostr, "SBI") = True Then
            Return 2
        ElseIf IsAsm(str, ostr, "ANI") = True Then
```

```
            Return 2
        ElseIf IsAsm(str, ostr, "XRI") = True Then
            Return 2
        ElseIf IsAsm(str, ostr, "ORI") = True Then
            Return 2
        ElseIf IsAsm(str, ostr, "CPI") = True Then
            Return 2
        ElseIf IsAsm(str, ostr, "JMP") = True Then
            Return 3
        ElseIf (IsAsm(str, ostr, "JNZ") = True) Or (IsAsm(str, ostr, "JZ") = True) Or _
               (IsAsm(str, ostr, "JNC") = True) Or (IsAsm(str, ostr, "JC") = True) Or _
               (IsAsm(str, ostr, "JPO") = True) Or (IsAsm(str, ostr, "JPE") = True) Or _
               (IsAsm(str, ostr, "JP") = True) Or (IsAsm(str, ostr, "JM") = True) _
               Then
            Return 3
        ElseIf IsAsm(str, ostr, "PCHL") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "CALL") = True Then
            Return 3
        ElseIf (IsAsm(str, ostr, "CNZ") = True) Or (IsAsm(str, ostr, "CZ") = True) Or _
               (IsAsm(str, ostr, "CNC") = True) Or (IsAsm(str, ostr, "CC") = True) Or _
               (IsAsm(str, ostr, "CPO") = True) Or (IsAsm(str, ostr, "CPE") = True) Or _
               (IsAsm(str, ostr, "CP") = True) Or (IsAsm(str, ostr, "CM") = True) _
               Then
            Return 3
        ElseIf IsAsm(str, ostr, "RET") = True Then
            Return 1
        ElseIf (IsAsm(str, ostr, "RNZ") = True) Or (IsAsm(str, ostr, "RZ") = True) Or _
               (IsAsm(str, ostr, "RNC") = True) Or (IsAsm(str, ostr, "RC") = True) Or _
               (IsAsm(str, ostr, "RPO") = True) Or (IsAsm(str, ostr, "RPE") = True) Or _
               (IsAsm(str, ostr, "RP") = True) Or (IsAsm(str, ostr, "RM") = True) _
               Then
            Return 1
        ElseIf IsAsm(str, ostr, "RST") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "PUSH") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "POP") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "XTHL") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "SPHL") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "OUT") = True Then
            Return 2
        ElseIf IsAsm(str, ostr, "IN") = True Then
            Return 2
        ElseIf IsAsm(str, ostr, "DI") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "EI") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "NOP") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "HLT") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "RIM") = True Then
            Return 1
        ElseIf IsAsm(str, ostr, "SIM") = True Then
            Return 1
        End If
        Return 0
    End Function

    Private Sub TrimProg(ByVal prog As String, ByRef ll As List(Of String))
        WriteLine("Program is being trimmed for removal of spaces...")
        While Not (prog.Trim.Length < 1)
            Dim eol As Integer
            eol = prog.IndexOf(vbNewLine)
            If eol = -1 Then
                eol = prog.IndexOf(vbCr)
            End If
```

```
            If eol = -1 Then
                eol = prog.IndexOf(vbLf)
            End If
            If eol = -1 Then
                eol = prog.Length
            End If


            Dim ln As String
            ln = prog.Substring(0, eol)
            If ((eol + 1) < prog.Length) Then
                prog = prog.Substring(eol + 1)
            Else
                prog = ""
            End If

            ln = ln.Trim
            If Not (ln.Length < 1) Then
                ll.Add(ln)
            End If
        End While

        WriteLine("Program trimmed successfully")
    End Sub
    Private Function AsmPreprocess(ByVal prog As String) As String
        WriteLine("Prepsocessing Phase 1 started...")
        Dim sb As New StringBuilder()
        Dim n As Integer = 0
        Dim eol As Integer
        eol = 0

        While prog.Length > 0
            eol = prog.IndexOf(vbNewLine)
            If eol = -1 Then
                eol = prog.IndexOf(vbCr)
            End If
            If eol = -1 Then
                eol = prog.IndexOf(vbLf)
            End If
            If eol = -1 Then
                eol = prog.Length
            End If


            Dim ln As String
            ln = prog.Substring(0, eol)
            If ((eol + 1) < prog.Length) Then
                prog = prog.Substring(eol + 1)
            Else
                prog = ""
            End If


            If ln.Trim.Length < 1 Then
                Continue While
            ElseIf ln.Trim.IndexOf(";") = 0 Then
                Continue While
            ElseIf Not (ln.IndexOf(";") = -1) Then
                n += 1
                sb.Append(ln.IndexOf(";") - 1)
                If ln.IndexOf(vbNewLine) = -1 Then
                    sb.Append(vbNewLine)
                End If
                Continue While
            ElseIf ln.Trim.ToUpper.IndexOf(".INCLUDE") = 0 Then
                n += 1
                IncludeFile(ln.Trim.Substring(8), sb)
                Continue While
            ElseIf ln.Trim.ToUpper.IndexOf("INCLUDE") = 0 Then
                n += 1
                IncludeFile(ln.Trim.Substring(7), sb)
```

```
                Continue While
            Else
                sb.Append(ln)
                If ln.IndexOf(vbNewLine) = -1 Then
                    sb.Append(vbNewLine)
                End If
            End If
        End While

        WriteLine("Preprocessing Phase 1 ended successfully.")

        If n < 1 Then
            Return sb.ToString()
        Else
            Return AsmPreprocess(sb.ToString())
        End If
    End Function
    Private Sub IncludeFile(ByVal ln As String, ByRef sb As StringBuilder)
        If ln.IndexOf(Chr(34)) = -1 Then
            Throw New Exception("File format specification invalid in " + ln)
        End If
        ln = ln.Substring(ln.IndexOf(Chr(34)) + 1)

        If ln.IndexOf(Chr(34)) = -1 Then
            Throw New Exception("File format specification invalid ending in " + ln)
        End If
        ln = ln.Substring(0, ln.IndexOf(Chr(34)))

        WriteLine(String.Format("Requested Inclusion of file {0}", ln))

        Dim fif As String = ""
        Dim fil As New FileInfo(ln)
        If fil.Exists = True Then
            fif = fil.FullName
        End If
        If fif.Trim.Length < 1 Then
            If fname.Trim.Length > 0 Then
                Dim cw As New FileInfo(fname)
                Dim temp As New FileInfo(cw.DirectoryName + "/" + fil.Name)
                If temp.Exists = True Then
                    fif = temp.FullName
                End If
            End If
        End If
        Dim fexe As New FileInfo(System.Diagnostics.Process.GetCurrentProcess().MainModule.FileName)
        Dim dexe As New DirectoryInfo(fexe.DirectoryName)
        Dim dinc As New DirectoryInfo(fexe.DirectoryName + "/include")
        Try
            If dinc.Exists = False Then
                dinc.Create()
            End If
        Catch ex As Exception

        End Try

        If fif.Trim.Length < 1 Then
            Dim temp As New FileInfo(dinc.FullName + "/" + fil.Name)
            If temp.Exists = True Then
                fif = temp.FullName
            End If
        End If

        If fif.Trim.Length < 1 Then
            Dim temp As New FileInfo(dexe.FullName + "/" + fil.Name)
            If temp.Exists = True Then
                fif = temp.FullName
            End If
        End If

        If fif.Trim.Length < 1 Then
            Throw New Exception("File " + ln + " not found")
```

```
        End If

        WriteLine("File requested found at " + fif)

        Dim sr As New StreamReader(fif)
        sb.Append(sr.ReadToEnd())
        sr.Close()
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    Timer1.Tick
        Dim str As String
        str = Me.Text
        Dim idx As Integer
        idx = str.IndexOf("*")
        Dim bmod As Boolean = False
        If bDirty = True Then
            If idx = -1 Then
                str += "*"
                bmod = True
            End If
        Else
            If idx <> -1 Then
                Dim s1, s2 As String
                s1 = ""
                s2 = ""
                Try
                    s1 = str.Substring(0, idx)
                    s2 = str.Substring(idx + 1)
                Catch ex As Exception

                End Try
                str = s1 + s2
                bmod = True
            End If
        End If

        If bmod = True Then
            Me.Text = str
        End If
    End Sub
End Class


c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormBitDisplay
    .Designer.vb
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormBitDisplay
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
```

```
        Me.ck7 = New System.Windows.Forms.CheckBox()
        Me.ck6 = New System.Windows.Forms.CheckBox()
        Me.ck5 = New System.Windows.Forms.CheckBox()
        Me.ck4 = New System.Windows.Forms.CheckBox()
        Me.ck3 = New System.Windows.Forms.CheckBox()
        Me.ck2 = New System.Windows.Forms.CheckBox()
        Me.ck1 = New System.Windows.Forms.CheckBox()
        Me.ck0 = New System.Windows.Forms.CheckBox()
        Me.SuspendLayout()
        '
        'ck7
        '
        Me.ck7.AutoSize = True
        Me.ck7.Location = New System.Drawing.Point(12, 12)
        Me.ck7.Name = "ck7"
        Me.ck7.Size = New System.Drawing.Size(79, 26)
        Me.ck7.TabIndex = 0
        Me.ck7.Text = "BIT 7"
        Me.ck7.UseVisualStyleBackColor = True
        '
        'ck6
        '
        Me.ck6.AutoSize = True
        Me.ck6.Location = New System.Drawing.Point(97, 12)
        Me.ck6.Name = "ck6"
        Me.ck6.Size = New System.Drawing.Size(79, 26)
        Me.ck6.TabIndex = 1
        Me.ck6.Text = "BIT 6"
        Me.ck6.UseVisualStyleBackColor = True
        '
        'ck5
        '
        Me.ck5.AutoSize = True
        Me.ck5.Location = New System.Drawing.Point(182, 12)
        Me.ck5.Name = "ck5"
        Me.ck5.Size = New System.Drawing.Size(79, 26)
        Me.ck5.TabIndex = 2
        Me.ck5.Text = "BIT 5"
        Me.ck5.UseVisualStyleBackColor = True
        '
        'ck4
        '
        Me.ck4.AutoSize = True
        Me.ck4.Location = New System.Drawing.Point(267, 12)
        Me.ck4.Name = "ck4"
        Me.ck4.Size = New System.Drawing.Size(79, 26)
        Me.ck4.TabIndex = 3
        Me.ck4.Text = "BIT 4"
        Me.ck4.UseVisualStyleBackColor = True
        '
        'ck3
        '
        Me.ck3.AutoSize = True
        Me.ck3.Location = New System.Drawing.Point(352, 12)
        Me.ck3.Name = "ck3"
        Me.ck3.Size = New System.Drawing.Size(79, 26)
        Me.ck3.TabIndex = 4
        Me.ck3.Text = "BIT 3"
        Me.ck3.UseVisualStyleBackColor = True
        '
        'ck2
        '
        Me.ck2.AutoSize = True
        Me.ck2.Location = New System.Drawing.Point(437, 12)
        Me.ck2.Name = "ck2"
        Me.ck2.Size = New System.Drawing.Size(79, 26)
        Me.ck2.TabIndex = 5
        Me.ck2.Text = "BIT 2"
        Me.ck2.UseVisualStyleBackColor = True
        '
        'ck1
```

```
        '
        Me.ck1.AutoSize = True
        Me.ck1.Location = New System.Drawing.Point(522, 12)
        Me.ck1.Name = "ck1"
        Me.ck1.Size = New System.Drawing.Size(79, 26)
        Me.ck1.TabIndex = 6
        Me.ck1.Text = "BIT 1"
        Me.ck1.UseVisualStyleBackColor = True
        '
        'ck0
        '
        Me.ck0.AutoSize = True
        Me.ck0.Location = New System.Drawing.Point(607, 12)
        Me.ck0.Name = "ck0"
        Me.ck0.Size = New System.Drawing.Size(79, 26)
        Me.ck0.TabIndex = 7
        Me.ck0.Text = "BIT 0"
        Me.ck0.UseVisualStyleBackColor = True
        '
        'FormBitDisplay
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(10.0!, 22.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.BackColor = System.Drawing.Color.Navy
        Me.ClientSize = New System.Drawing.Size(689, 51)
        Me.Controls.Add(Me.ck0)
        Me.Controls.Add(Me.ck1)
        Me.Controls.Add(Me.ck2)
        Me.Controls.Add(Me.ck3)
        Me.Controls.Add(Me.ck4)
        Me.Controls.Add(Me.ck5)
        Me.Controls.Add(Me.ck6)
        Me.Controls.Add(Me.ck7)
        Me.Font = New System.Drawing.Font("Consolas", 14.25!, System.Drawing.FontStyle.Bold, System. ↙
    Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.ForeColor = System.Drawing.Color.FromArgb(CType(CType(255, Byte), Integer), CType(CType    ↙
    (255, Byte), Integer), CType(CType(128, Byte), Integer))
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog
        Me.Margin = New System.Windows.Forms.Padding(5)
        Me.MaximizeBox = False
        Me.Name = "FormBitDisplay"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Bit Display [IN=03H, OUT=03H PORT]"
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents ck7 As System.Windows.Forms.CheckBox
    Friend WithEvents ck6 As System.Windows.Forms.CheckBox
    Friend WithEvents ck5 As System.Windows.Forms.CheckBox
    Friend WithEvents ck4 As System.Windows.Forms.CheckBox
    Friend WithEvents ck3 As System.Windows.Forms.CheckBox
    Friend WithEvents ck2 As System.Windows.Forms.CheckBox
    Friend WithEvents ck1 As System.Windows.Forms.CheckBox
    Friend WithEvents ck0 As System.Windows.Forms.CheckBox
End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormBitDisplay ↙
    .vb
********************************************************************************
Public Class FormBitDisplay

    Private Sub FormBitDisplay_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)      ↙
    Handles MyBase.Load

    End Sub

    Private Sub CheckBoxState(ByVal i As Integer, ByVal bs As Integer, ByVal cks As CheckBox)
        Dim ib As Integer
```

```
        ib = i >> bs
        ib = ib And &H1
        If ib = 1 Then
            cks.Checked = True
            cks.ForeColor = Color.LightYellow
        Else
            cks.Checked = False
            cks.ForeColor = Color.LimeGreen
        End If
    End Sub
    Private Sub SetInteger(ByVal i As Integer)
        FormMain.MakeMeFirst(Me)
        CheckBoxState(i, 0, ck0)
        CheckBoxState(i, 1, ck1)
        CheckBoxState(i, 2, ck2)
        CheckBoxState(i, 3, ck3)
        CheckBoxState(i, 4, ck4)
        CheckBoxState(i, 5, ck5)
        CheckBoxState(i, 6, ck6)
        CheckBoxState(i, 7, ck7)
    End Sub

    Private Sub IntFromCheck(ByRef i As Integer, ByVal bs As Integer, ByVal cks As CheckBox)
        If cks.Checked = True Then
            i = i Or (1 << bs)
        End If
    End Sub
    Private Function GetInteger() As Integer
        FormMain.MakeMeFirst(Me)
        Dim i As Integer = 0
        IntFromCheck(i, 0, ck0)
        IntFromCheck(i, 1, ck1)
        IntFromCheck(i, 2, ck2)
        IntFromCheck(i, 3, ck3)
        IntFromCheck(i, 4, ck4)
        IntFromCheck(i, 5, ck5)
        IntFromCheck(i, 6, ck6)
        IntFromCheck(i, 7, ck7)
        Return i
    End Function


    Private Shared kd As FormBitDisplay = Nothing
    Public Shared Sub LoadMe(ByVal p As FormMain)
        If kd Is Nothing Then
            kd = New FormBitDisplay()
            kd.MdiParent = p
            kd.Show()
        End If
    End Sub

    Public Shared Sub OutPort(ByVal p As FormMain, ByVal ch As Integer)
        Try
            LoadMe(p)
            kd.SetInteger(ch)
        Catch ex As Exception
            FormReport.WriteLine("Exception in FormKeyDisplay.OutPort(...): " + ex.Message)
        End Try

    End Sub
    Public Shared Function InPort(ByVal p As FormMain) As Integer
        Try
            LoadMe(p)
            Return kd.GetInteger()
        Catch ex As Exception

            Return 0
        End Try
    End Function

    Private Sub FormBitDisplay_FormClosing(ByVal sender As System.Object, ByVal e As System.Windows. ↙
```

```
    Forms.FormClosingEventArgs) Handles MyBase.FormClosing
        kd = Nothing
    End Sub
End Class


c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormDecDisplay↙
    .Designer.vb
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormDecDisplay
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.Label1 = New System.Windows.Forms.Label()
        Me.tbInput = New System.Windows.Forms.TextBox()
        Me.Label2 = New System.Windows.Forms.Label()
        Me.lbOutput = New System.Windows.Forms.Label()
        Me.SuspendLayout()
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Location = New System.Drawing.Point(12, 22)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(210, 22)
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "Enter Decimal Value:"
        '
        'tbInput
        '
        Me.tbInput.BackColor = System.Drawing.Color.FromArgb(CType(CType(0, Byte), Integer), CType    ↙
    (CType(0, Byte), Integer), CType(CType(192, Byte), Integer))
        Me.tbInput.ForeColor = System.Drawing.Color.Yellow
        Me.tbInput.Location = New System.Drawing.Point(251, 19)
        Me.tbInput.Name = "tbInput"
        Me.tbInput.Size = New System.Drawing.Size(100, 30)
        Me.tbInput.TabIndex = 1
        '
        'Label2
        '
        Me.Label2.AutoSize = True
        Me.Label2.Location = New System.Drawing.Point(12, 72)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(80, 22)
        Me.Label2.TabIndex = 2
        Me.Label2.Text = "Output:"
        '
        'lbOutput
        '
        Me.lbOutput.AutoSize = True
        Me.lbOutput.Location = New System.Drawing.Point(247, 72)
```

```
        Me.lbOutput.Name = "lbOutput"
        Me.lbOutput.Size = New System.Drawing.Size(20, 22)
        Me.lbOutput.TabIndex = 3
        Me.lbOutput.Text = "0"
        '
        'FormDecDisplay
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(10.0!, 22.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.BackColor = System.Drawing.Color.FromArgb(CType(CType(0, Byte), Integer), CType(CType(0,
    Byte), Integer), CType(CType(192, Byte), Integer))
        Me.ClientSize = New System.Drawing.Size(391, 117)
        Me.Controls.Add(Me.lbOutput)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.tbInput)
        Me.Controls.Add(Me.Label1)
        Me.Font = New System.Drawing.Font("Consolas", 14.25!, System.Drawing.FontStyle.Regular,
    System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.ForeColor = System.Drawing.Color.Yellow
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog
        Me.Margin = New System.Windows.Forms.Padding(5)
        Me.MaximizeBox = False
        Me.Name = "FormDecDisplay"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Decimal IO [IN=PORT 02H, OUT=PORT 02H]"
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents tbInput As System.Windows.Forms.TextBox
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents lbOutput As System.Windows.Forms.Label
End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormDecDisplay
    .vb
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Imports System.Threading
Public Class FormDecDisplay
    Private ev As New ManualResetEvent(False)
    Private Sub FormDecDisplay_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
        ev.Reset()
    End Sub



    Private Sub SetInteger(ByVal i As Integer)
        FormMain.MakeMeFirst(Me)

        Dim sign As Integer = i >> 7
        sign = sign And &H1
        If sign = 1 Then
            i = Not (i)
            i = i And &H7F
            i = i + 1
            i = i And &HFF
            i = i * -1
        End If
        lbOutput.Text = i.ToString()
    End Sub

    Private Function GetInteger() As Integer
        FormMain.MakeMeFirst(Me)
        Dim i As Integer = 0
        Try
            i = Integer.Parse(tbInput.Text, Globalization.NumberStyles.Integer)
            If i < 0 Then
```

```
                i = i And &HFF
            Else
                i = i And &H7F
            End If

        Catch ex As Exception

        End Try
        tbInput.Clear()
        Return i
    End Function

    Private Shared kd As FormDecDisplay = Nothing
    Public Shared Sub LoadMe(ByVal p As FormMain)
        If kd Is Nothing Then
            kd = New FormDecDisplay()
            kd.MdiParent = p
            kd.Show()
        End If
    End Sub


    Public Shared Sub OutPort(ByVal p As FormMain, ByVal ch As Integer)
        Try
            LoadMe(p)
            kd.SetInteger(ch)
        Catch ex As Exception
            FormReport.WriteLine("Exception in FormKeyDisplay.OutPort(...): " + ex.Message)
        End Try

    End Sub
    Public Shared Function InPort(ByVal p As FormMain) As Integer
        Try
            LoadMe(p)
            Return kd.GetInteger()
        Catch ex As Exception

            Return 0
        End Try
    End Function


    Private Sub FormDecDisplay_FormClosing(ByVal sender As System.Object, ByVal e As System.Windows. ↙
    Forms.FormClosingEventArgs) Handles MyBase.FormClosing
        kd = Nothing
    End Sub


    Private Sub tbInput_KeyUp(ByVal sender As System.Object, ByVal e As System.Windows.Forms.        ↙
    KeyEventArgs) Handles tbInput.KeyUp
        If e.KeyCode = Keys.Enter Then
            Try
                Dim i As Integer = Integer.Parse(tbInput.Text, Globalization.NumberStyles.Integer)
                If i < 0 Then
                    i = i And &HFF
                Else
                    i = i And &H7F
                End If

                Dim sign As Integer = i >> 7
                sign = sign And &H1
                If sign = 1 Then
                    i = Not (i)
                    i = i And &H7F
                    i = i + 1
                    i = i And &HFF
                    i = i * -1
                End If
                tbInput.Text = i.ToString()
                ev.Set()
            Catch ex As Exception
```

```
                tbInput.Text = ""
            End Try
        End If
    End Sub


    Public Shared Function PortInWait() As Boolean
        If kd Is Nothing Then
            Return False
        End If
        While kd.ev.WaitOne() <> True
            Thread.Sleep(50)
        End While
        kd.ev.Reset()
        Return True
    End Function

End Class


c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormHexDecode.↙
    Designer.vb
*******************************************************************************
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormHexDecode
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.rtbSrc = New System.Windows.Forms.RichTextBox()
        Me.btnLoad = New System.Windows.Forms.Button()
        Me.wbDsc = New System.Windows.Forms.WebBrowser()
        Me.SuspendLayout()
        '
        'rtbSrc
        '
        Me.rtbSrc.Location = New System.Drawing.Point(14, 13)
        Me.rtbSrc.Name = "rtbSrc"
        Me.rtbSrc.Size = New System.Drawing.Size(564, 116)
        Me.rtbSrc.TabIndex = 0
        Me.rtbSrc.Text = ""
        '
        'btnLoad
        '
        Me.btnLoad.Location = New System.Drawing.Point(588, 14)
        Me.btnLoad.Name = "btnLoad"
        Me.btnLoad.Size = New System.Drawing.Size(72, 114)
        Me.btnLoad.TabIndex = 1
        Me.btnLoad.Text = "Load"
        Me.btnLoad.UseVisualStyleBackColor = True
        '
        'wbDsc
        '
```

```
        Me.wbDsc.IsWebBrowserContextMenuEnabled = False
        Me.wbDsc.Location = New System.Drawing.Point(16, 137)
        Me.wbDsc.MinimumSize = New System.Drawing.Size(20, 20)
        Me.wbDsc.Name = "wbDsc"
        Me.wbDsc.Size = New System.Drawing.Size(643, 263)
        Me.wbDsc.TabIndex = 2
        '
        'FormHexDecode
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(667, 412)
        Me.Controls.Add(Me.wbDsc)
        Me.Controls.Add(Me.btnLoad)
        Me.Controls.Add(Me.rtbSrc)
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog
        Me.MaximizeBox = False
        Me.Name = "FormHexDecode"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Intel HEX Decoder"
        Me.ResumeLayout(False)

    End Sub
    Friend WithEvents rtbSrc As System.Windows.Forms.RichTextBox
    Friend WithEvents btnLoad As System.Windows.Forms.Button
    Friend WithEvents wbDsc As System.Windows.Forms.WebBrowser
End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormHexDecode. ↙
    vb
********************************************************************************
Imports System.Text
Public Class FormHexDecode

    Private Sub FormHexDecode_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)       ↙
    Handles MyBase.Load
        rtbSrc.Focus()
    End Sub

    Private Sub btnLoad_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles    ↙
    btnLoad.Click
        Try
            Dim frm As New OpenFileDialog()
            frm.Filter = "Intel HEX file (*.hex)|*.hex|All files(*.*)|*.*||"
            frm.Multiselect = False
            frm.Title = "Load an Intel HEX file to display content"
            If frm.ShowDialog() = DialogResult.OK Then
                Dim fl As New System.IO.StreamReader(frm.FileName)
                rtbSrc.Text = fl.ReadToEnd()
                fl.Close()
            End If
            rtbSrc.Focus()
        Catch ex As Exception
            Me.Dispose()
        End Try

    End Sub

    Private Sub rtbSrc_KeyUp(ByVal sender As Object, ByVal e As System.Windows.Forms.KeyEventArgs)   ↙
    Handles rtbSrc.KeyUp
        If e.KeyCode = Keys.Enter Then
            Try
                DecodeHex()
            Catch ex As Exception
                Dim s As String
                s = wbDsc.DocumentText
                s += String.Format("<hr><br><tt><font color=blue>Exception: {0}</font></tt><br><br>  ↙
    <hr>", ex.Message)
                wbDsc.DocumentText = s
```

```
            End Try
        End If
    End Sub

    Private Sub DecodeHex()
        Dim str As String
        Dim s As New StringBuilder()
        str = rtbSrc.Text
        s.Append("<html>")
        s.Append("<head><title>HEX Decoder</title></head>")
        s.Append("<body>")

        s.Append("<table cellpadding=5 cellspacing=0 border=1 align=center width='90%'>")
        s.Append("<tr>")
        s.Append("<th>Address</th>")
        s.Append("<th>Byte Value</th>")
        s.Append("</tr>")

        Dim i As Integer = 0
        Dim addr As Integer = 0
        Dim bt As Integer = 0

        While i < str.Length
            If str.Substring(i, 1) = ":" Then
                i += 1
                Dim recLen As Integer
                recLen = Integer.Parse(str.Substring(i, 2), Globalization.NumberStyles.HexNumber) And ↙
     &HFF
                i += 2

                Dim offset As Integer
                offset = Integer.Parse(str.Substring(i, 4), Globalization.NumberStyles.HexNumber) And ↙
     &HFFFF
                i += 4

                Dim recType As Integer
                recType = Integer.Parse(str.Substring(i, 2), Globalization.NumberStyles.HexNumber)    ↙
    And &HFF
                i += 2

                addr = offset

                If recType = 1 Then
                    Exit While
                End If

                For j As Integer = 0 To recLen
                    bt = Integer.Parse(str.Substring(i, 2), Globalization.NumberStyles.HexNumber)
                    i += 2
                    If (str.Substring(i, 1) = vbCr) Or (str.Substring(i, 1) = vbLf) Then
                        Exit For
                    End If

                    s.Append("<tr>")
                    s.Append(String.Format("<td align=center><font color=blue family='Consolas'>{0:   ↙
    X4}</font></td><td align=center><font color=borwn family='Consolas'>{1:X2}</font></td>", addr,    ↙
    bt))
                    s.Append("</tr>")
                    addr += 1
                Next
            End If
            i += 1
        End While

        s.Append("</table>")
        s.Append("</body></html>")
        wbDsc.DocumentText = s.ToString()
    End Sub
End Class
```

```
c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormHexEncode.↙
    Designer.vb
*******************************************************************************
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormHexEncode
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.Label1 = New System.Windows.Forms.Label()
        Me.tbStart = New System.Windows.Forms.TextBox()
        Me.tbSrc = New System.Windows.Forms.RichTextBox()
        Me.wbDst = New System.Windows.Forms.WebBrowser()
        Me.SuspendLayout()
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Location = New System.Drawing.Point(12, 28)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(73, 13)
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "Start Address:"
        '
        'tbStart
        '
        Me.tbStart.Location = New System.Drawing.Point(91, 25)
        Me.tbStart.Name = "tbStart"
        Me.tbStart.Size = New System.Drawing.Size(83, 20)
        Me.tbStart.TabIndex = 1
        '
        'tbSrc
        '
        Me.tbSrc.Location = New System.Drawing.Point(12, 51)
        Me.tbSrc.Name = "tbSrc"
        Me.tbSrc.Size = New System.Drawing.Size(538, 141)
        Me.tbSrc.TabIndex = 2
        Me.tbSrc.Text = ""
        '
        'wbDst
        '
        Me.wbDst.IsWebBrowserContextMenuEnabled = False
        Me.wbDst.Location = New System.Drawing.Point(15, 204)
        Me.wbDst.MinimumSize = New System.Drawing.Size(20, 20)
        Me.wbDst.Name = "wbDst"
        Me.wbDst.Size = New System.Drawing.Size(534, 209)
        Me.wbDst.TabIndex = 3
        '
        'FormHexEncode
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(562, 425)
```

```
        Me.Controls.Add(Me.wbDst)
        Me.Controls.Add(Me.tbSrc)
        Me.Controls.Add(Me.tbStart)
        Me.Controls.Add(Me.Label1)
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog
        Me.MaximizeBox = False
        Me.Name = "FormHexEncode"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Hex Encoder"
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents tbStart As System.Windows.Forms.TextBox
    Friend WithEvents tbSrc As System.Windows.Forms.RichTextBox
    Friend WithEvents wbDst As System.Windows.Forms.WebBrowser
End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormHexEncode. ↙
    vb
********************************************************************************
Imports System.Text
Public Class FormHexEncode

    Private Sub FormHexEncode_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)       ↙
    Handles MyBase.Load
        tbStart.Text = "0000"
        tbSrc.Focus()
    End Sub

    Private Sub tbSrc_KeyUp(ByVal sender As Object, ByVal e As System.Windows.Forms.KeyEventArgs)    ↙
    Handles tbSrc.KeyUp
        If e.KeyCode = Keys.Enter Then
            Try
                Dim sa As Integer
                sa = Integer.Parse(tbStart.Text, Globalization.NumberStyles.HexNumber)
                sa = sa And &HFFFF
                Dim src As String
                src = tbSrc.Text
                EncodeHex(sa, src)
            Catch ex As Exception
                wbDst.DocumentText += String.Format("<br><br><hr><br><b><tt>Exception: <font color=  ↙
    blue>{0}</font></tt></b><br><br><hr>", ex.Message)
            End Try
        End If
    End Sub


    Private Sub EncodeHex(ByVal sa As Integer, ByVal src As String)
        Dim db As New StringBuilder()
        db.Append("<html><head><title>HEX Encoder</title></head><body>")
        db.Append("<table cellspacing=0 cellpadding=5 border=2 width='90%' align=center>")
        db.Append("<tr><th>HEX Data</th></tr>")
        db.Append("<tr><td><font color=blue family='Consolas'>")
        Dim i As Integer
        i = 0
        Dim ll As New List(Of Integer)
        While i < src.Length
            If Not (src.Substring(i, 1).Trim.Length < 1) Then
                Dim dataByte As Integer
                Try
                    dataByte = Integer.Parse(src.Substring(i, Math.Min(2, src.Length - i)),          ↙
    Globalization.NumberStyles.HexNumber)
                    i += Math.Min(2, src.Length - i)
                    ll.Add(dataByte)
                    Continue While
                Catch ex As Exception
```

```
                End Try
            End If
            i += 1
        End While

        Dim arr() As Integer = ll.ToArray()
        Dim mkr As Integer = 0
        While mkr < arr.Length
            db.Append(":")

            Dim sdt As New StringBuilder()
            Dim startaddress As Integer = sa
            Dim ck As Integer = 0
            For mm As Integer = 0 To &HF
                sdt.Append(String.Format("{0:X2}", arr(mkr)))
                ck += arr(mkr)
                sa += 1
                mkr += 1
                If Not (mkr < arr.Length) Then
                    Exit For
                End If
            Next
            Dim sdt_len As Integer
            sdt_len = sdt.Length
            sdt_len = sdt_len / 2
            ck += sdt_len
            ck += startaddress And &HFF
            ck += (startaddress >> 8) And &HFF
            ck = Not (ck)
            ck += 1
            ck = ck And &HFF
            db.Append(String.Format("{0:X2}{1:X4}00{2}{3:X2}", sdt_len, startaddress, sdt.ToString(), ck))
            db.Append("<br>")
        End While
        db.Append("</td></tr>")
        db.Append("<tr><th>End of HEX File Marker</th></tr>")
        db.Append("<tr><td><font color=blue family='Consolas'>:00000001FF</font></td></tr>")
        db.Append("</table>")
        db.Append("</body></html>")

        wbDst.DocumentText = db.ToString()
    End Sub
End Class


c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormHexIO.Designer.vb
********************************************************************************
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormHexIO
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
```

```
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.Label1 = New System.Windows.Forms.Label()
        Me.tbInput = New System.Windows.Forms.TextBox()
        Me.Label2 = New System.Windows.Forms.Label()
        Me.lbOutput = New System.Windows.Forms.Label()
        Me.SuspendLayout()
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Font = New System.Drawing.Font("Consolas", 14.25!, System.Drawing.FontStyle.Regular
    , System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label1.ForeColor = System.Drawing.Color.Yellow
        Me.Label1.Location = New System.Drawing.Point(12, 9)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(130, 22)
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "Enter Value:"
        '
        'tbInput
        '
        Me.tbInput.BackColor = System.Drawing.Color.FromArgb(CType(CType(0, Byte), Integer), CType
    (CType(0, Byte), Integer), CType(CType(192, Byte), Integer))
        Me.tbInput.Font = New System.Drawing.Font("Consolas", 14.25!, System.Drawing.FontStyle.
    Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.tbInput.ForeColor = System.Drawing.Color.FromArgb(CType(CType(255, Byte), Integer), CType
    (CType(255, Byte), Integer), CType(CType(192, Byte), Integer))
        Me.tbInput.Location = New System.Drawing.Point(160, 6)
        Me.tbInput.MaxLength = 4
        Me.tbInput.Name = "tbInput"
        Me.tbInput.Size = New System.Drawing.Size(74, 30)
        Me.tbInput.TabIndex = 1
        '
        'Label2
        '
        Me.Label2.AutoSize = True
        Me.Label2.Font = New System.Drawing.Font("Consolas", 14.25!, System.Drawing.FontStyle.Regular
    , System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label2.ForeColor = System.Drawing.Color.Yellow
        Me.Label2.Location = New System.Drawing.Point(12, 59)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(80, 22)
        Me.Label2.TabIndex = 3
        Me.Label2.Text = "Output:"
        '
        'lbOutput
        '
        Me.lbOutput.AutoSize = True
        Me.lbOutput.Font = New System.Drawing.Font("Consolas", 14.25!, System.Drawing.FontStyle.
    Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.lbOutput.ForeColor = System.Drawing.Color.Yellow
        Me.lbOutput.Location = New System.Drawing.Point(156, 59)
        Me.lbOutput.Name = "lbOutput"
        Me.lbOutput.Size = New System.Drawing.Size(30, 22)
        Me.lbOutput.TabIndex = 4
        Me.lbOutput.Text = "00"
        '
        'FormHexIO
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.BackColor = System.Drawing.Color.FromArgb(CType(CType(0, Byte), Integer), CType(CType(0,
    Byte), Integer), CType(CType(192, Byte), Integer))
        Me.ClientSize = New System.Drawing.Size(382, 96)
        Me.Controls.Add(Me.lbOutput)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.tbInput)
        Me.Controls.Add(Me.Label1)
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog
        Me.MaximizeBox = False
```

```
        Me.Name = "FormHexIO"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Hexadecimal IO [IN: 01H PORT OUT: 01H PORT]"
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents tbInput As System.Windows.Forms.TextBox
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents lbOutput As System.Windows.Forms.Label
End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormHexIO.vb
*******************************************************************************
Imports System.Threading
Public Class FormHexIO
    Private ev As New ManualResetEvent(False)

    Private Sub FormHexIO_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles    ↙
    MyBase.Load
        tbInput.Text = ""
        kd.ev.Reset()
    End Sub

    Private Sub SetInteger(ByVal ival As Integer)
        FormMain.MakeMeFirst(Me)
        lbOutput.Text = String.Format("{0:X2}", ival)
    End Sub

    Private Function GetInteger() As Integer
        FormMain.MakeMeFirst(Me)
        Dim i As Integer = 0
        Try
            i = Integer.Parse(tbInput.Text, Globalization.NumberStyles.HexNumber)
        Catch ex As Exception

        End Try
        tbInput.Clear()
        Return i
    End Function


    Public Shared Function PortInWait() As Boolean
        If kd Is Nothing Then
            Return False
        End If
        While kd.ev.WaitOne() <> True
            Thread.Sleep(50)
        End While
        kd.ev.Reset()
        Return True
    End Function



    Private Shared kd As FormHexIO = Nothing
    Public Shared Sub LoadMe(ByVal p As FormMain)
        If kd Is Nothing Then
            kd = New FormHexIO()
            kd.MdiParent = p
            kd.Show()
        End If
    End Sub

    Public Shared Sub OutPort(ByVal p As FormMain, ByVal ch As Integer)
        Try
            LoadMe(p)
            kd.SetInteger(ch)
```

```
        Catch ex As Exception
            FormReport.WriteLine("Exception in FormKeyDisplay.OutPort(...): " + ex.Message)
        End Try

    End Sub
    Public Shared Function InPort(ByVal p As FormMain) As Integer
        Try
            LoadMe(p)
            Return kd.GetInteger()
        Catch ex As Exception

            Return 0
        End Try
    End Function


    Private Sub FormHexIO_FormClosing(ByVal sender As System.Object, ByVal e As System.Windows.Forms. ↙
    FormClosingEventArgs) Handles MyBase.FormClosing
        kd = Nothing
    End Sub

    Private Sub tbInput_KeyUp(ByVal sender As System.Object, ByVal e As System.Windows.Forms.        ↙
    KeyEventArgs) Handles tbInput.KeyUp
        If e.KeyCode = Keys.Enter Then
            Dim i As Integer = GetInteger()
            i = i And &HFF
            tbInput.Text = String.Format("{0:X2}", i)
            ev.Set()
        End If
    End Sub
End Class


c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormIntrSerial ↙
    .Designer.vb
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormIntrSerial
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.GroupBox1 = New System.Windows.Forms.GroupBox()
        Me.tbSod = New System.Windows.Forms.TextBox()
        Me.Label1 = New System.Windows.Forms.Label()
        Me.GroupBox2 = New System.Windows.Forms.GroupBox()
        Me.btnRst55 = New System.Windows.Forms.Button()
        Me.btnRst65 = New System.Windows.Forms.Button()
        Me.btnRst75 = New System.Windows.Forms.Button()
        Me.btnTrap = New System.Windows.Forms.Button()
        Me.btnSOD = New System.Windows.Forms.Button()
        Me.Label2 = New System.Windows.Forms.Label()
        Me.tbIntrFlag = New System.Windows.Forms.TextBox()
```

```
        Me.GroupBox1.SuspendLayout()
        Me.GroupBox2.SuspendLayout()
        Me.SuspendLayout()
        '
        'GroupBox1
        '
        Me.GroupBox1.Controls.Add(Me.tbIntrFlag)
        Me.GroupBox1.Controls.Add(Me.Label2)
        Me.GroupBox1.Controls.Add(Me.btnSOD)
        Me.GroupBox1.Controls.Add(Me.tbSod)
        Me.GroupBox1.Controls.Add(Me.Label1)
        Me.GroupBox1.Location = New System.Drawing.Point(12, 14)
        Me.GroupBox1.Name = "GroupBox1"
        Me.GroupBox1.Size = New System.Drawing.Size(405, 47)
        Me.GroupBox1.TabIndex = 0
        Me.GroupBox1.TabStop = False
        Me.GroupBox1.Text = "Serial Transmission:"
        '
        'tbSod
        '
        Me.tbSod.Location = New System.Drawing.Point(49, 19)
        Me.tbSod.MaxLength = 1
        Me.tbSod.Name = "tbSod"
        Me.tbSod.ReadOnly = True
        Me.tbSod.Size = New System.Drawing.Size(38, 20)
        Me.tbSod.TabIndex = 2
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Location = New System.Drawing.Point(15, 23)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(33, 13)
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "SOD:"
        '
        'GroupBox2
        '
        Me.GroupBox2.Controls.Add(Me.btnRst55)
        Me.GroupBox2.Controls.Add(Me.btnRst65)
        Me.GroupBox2.Controls.Add(Me.btnRst75)
        Me.GroupBox2.Controls.Add(Me.btnTrap)
        Me.GroupBox2.Location = New System.Drawing.Point(16, 82)
        Me.GroupBox2.Name = "GroupBox2"
        Me.GroupBox2.Size = New System.Drawing.Size(391, 182)
        Me.GroupBox2.TabIndex = 1
        Me.GroupBox2.TabStop = False
        Me.GroupBox2.Text = "Hardware Interrupts:"
        '
        'btnRst55
        '
        Me.btnRst55.Location = New System.Drawing.Point(95, 137)
        Me.btnRst55.Name = "btnRst55"
        Me.btnRst55.Size = New System.Drawing.Size(209, 27)
        Me.btnRst55.TabIndex = 3
        Me.btnRst55.Text = "RST 5.5"
        Me.btnRst55.UseVisualStyleBackColor = True
        '
        'btnRst65
        '
        Me.btnRst65.Location = New System.Drawing.Point(95, 104)
        Me.btnRst65.Name = "btnRst65"
        Me.btnRst65.Size = New System.Drawing.Size(209, 27)
        Me.btnRst65.TabIndex = 2
        Me.btnRst65.Text = "RST 6.5"
        Me.btnRst65.UseVisualStyleBackColor = True
        '
        'btnRst75
        '
        Me.btnRst75.Location = New System.Drawing.Point(95, 71)
        Me.btnRst75.Name = "btnRst75"
```

```
        Me.btnRst75.Size = New System.Drawing.Size(209, 27)
        Me.btnRst75.TabIndex = 1
        Me.btnRst75.Text = "RST 7.5"
        Me.btnRst75.UseVisualStyleBackColor = True
        '
        'btnTrap
        '
        Me.btnTrap.Location = New System.Drawing.Point(95, 28)
        Me.btnTrap.Name = "btnTrap"
        Me.btnTrap.Size = New System.Drawing.Size(209, 27)
        Me.btnTrap.TabIndex = 0
        Me.btnTrap.Text = "TRAP"
        Me.btnTrap.UseVisualStyleBackColor = True
        '
        'btnSOD
        '
        Me.btnSOD.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!, System.Drawing.
    FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.btnSOD.Location = New System.Drawing.Point(128, 18)
        Me.btnSOD.Name = "btnSOD"
        Me.btnSOD.Size = New System.Drawing.Size(75, 23)
        Me.btnSOD.TabIndex = 3
        Me.btnSOD.Text = "SID = 1"
        Me.btnSOD.UseVisualStyleBackColor = True
        '
        'Label2
        '
        Me.Label2.AutoSize = True
        Me.Label2.Location = New System.Drawing.Point(232, 22)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(72, 13)
        Me.Label2.TabIndex = 4
        Me.Label2.Text = "Interrupt Flag:"
        '
        'tbIntrFlag
        '
        Me.tbIntrFlag.Location = New System.Drawing.Point(325, 19)
        Me.tbIntrFlag.MaxLength = 1
        Me.tbIntrFlag.Name = "tbIntrFlag"
        Me.tbIntrFlag.ReadOnly = True
        Me.tbIntrFlag.Size = New System.Drawing.Size(38, 20)
        Me.tbIntrFlag.TabIndex = 5
        '
        'FormIntrSerial
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(424, 280)
        Me.Controls.Add(Me.GroupBox2)
        Me.Controls.Add(Me.GroupBox1)
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog
        Me.MaximizeBox = False
        Me.Name = "FormIntrSerial"
        Me.Text = "Interrupt Mask and Serial IO"
        Me.GroupBox1.ResumeLayout(False)
        Me.GroupBox1.PerformLayout()
        Me.GroupBox2.ResumeLayout(False)
        Me.ResumeLayout(False)

    End Sub
    Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
    Friend WithEvents tbSod As System.Windows.Forms.TextBox
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents GroupBox2 As System.Windows.Forms.GroupBox
    Friend WithEvents btnRst55 As System.Windows.Forms.Button
    Friend WithEvents btnRst65 As System.Windows.Forms.Button
    Friend WithEvents btnRst75 As System.Windows.Forms.Button
    Friend WithEvents btnTrap As System.Windows.Forms.Button
    Friend WithEvents btnSOD As System.Windows.Forms.Button
    Friend WithEvents tbIntrFlag As System.Windows.Forms.TextBox
    Friend WithEvents Label2 As System.Windows.Forms.Label
```

```
End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormIntrSerial.vb
********************************************************************************
Public Class FormIntrSerial
    Private clickStack As New Stack(Of Integer)
    Private Shared SerialPrio As Integer = 1

    Private Sub FormIntrSerial_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
        RemoveHandler FormMain.GetMachineState.OnEventPCUpdate, AddressOf OnEventPCUpdate
        RemoveHandler FormMain.GetMachineState.OnEventSID, AddressOf OnEventSID
        RemoveHandler FormMain.GetMachineState.OnEventSOD, AddressOf OnEventSOD
    End Sub

    Private Sub FormIntrSerial_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        RefreshSerial(FormMain.GetMachineState)
        RefreshIntr(FormMain.GetMachineState)
        AddHandler FormMain.GetMachineState.OnEventPCUpdate, AddressOf OnEventPCUpdate
        AddHandler FormMain.GetMachineState.OnEventSID, AddressOf OnEventSID
        AddHandler FormMain.GetMachineState.OnEventSOD, AddressOf OnEventSOD
    End Sub

    Private Sub RefreshSerial(ByVal m As MachineState)
        If m.IsSerialEnabled = True Then
            btnSOD.Enabled = True
        Else
            btnSOD.Enabled = False
        End If
    End Sub
    Private Sub RefreshIntr(ByVal m As MachineState)
        tbIntrFlag.Text = String.Format("{0:X2}", m.GetIntMask())
        If m.IsRST7_5Enabled = 1 Then
            btnRst75.Enabled = True
        Else
            btnRst75.Enabled = False
        End If

        If m.IsRST6_5Enabled = 1 Then
            btnRst65.Enabled = True
        Else
            btnRst65.Enabled = False
        End If

        If m.IsRST5_5Enabled = 1 Then
            btnRst55.Enabled = True
        Else
            btnRst55.Enabled = False
        End If
    End Sub

    Public Sub OnEventPCUpdate(ByVal pc As UInt16, ByRef macState As MachineState)
        RefreshIntrFlag(macState)
        RefreshIntr(macState)
    End Sub

    Private Sub btnTrap_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnTrap.Click
        RefreshIntrFlag(FormMain.GetMachineState())
        FormMain.GetMachineState.Interrupt_TRAP()
    End Sub

    Private Sub btnRst75_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnRst75.Click
        RefreshIntrFlag(FormMain.GetMachineState())
        FormMain.GetMachineState.Interrupt_RST7_5()
    End Sub
```

```
    Private Sub btnRst65_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↙
    btnRst65.Click
        RefreshIntrFlag(FormMain.GetMachineState())
        FormMain.GetMachineState.Interrupt_RST6_5()
    End Sub

    Private Sub btnRst55_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↙
    btnRst55.Click
        RefreshIntrFlag(FormMain.GetMachineState())
        FormMain.GetMachineState.Interrupt_RST5_5()
    End Sub

    Private Sub btnSOD_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↙
    btnSOD.Click
        RefreshIntrFlag(FormMain.GetMachineState())
        If FormMain.GetMachineState().IsRunning = True Then
            clickStack.Push(1)
        Else
            clickStack.Clear()
        End If
    End Sub

    Public Sub OnEventSID(ByVal bEnable As Boolean, ByRef val As Integer, ByRef prio As Integer, ↙
    ByVal macState As MachineState)
        RefreshIntrFlag(macState)
        btnSOD.Enabled = bEnable
        If SerialPrio > prio Then
            prio = SerialPrio
            Try
                If clickStack.Count < 1 Then
                    val = 0
                Else
                    val = clickStack.Pop() And &H1
                End If
            Catch ex As Exception
                val = 0
            End Try
        End If

    End Sub
    Public Sub OnEventSOD(ByVal bEnable As Boolean, ByRef val As Integer, ByVal macState As ↙
    MachineState)
        RefreshIntrFlag(macState)
        If bEnable = False Then
            Exit Sub
        End If

        tbSod.Text = String.Format("{0:X1}", val)
    End Sub

    Public Sub RefreshIntrFlag(ByVal m As MachineState)
        Dim str As String
        str = String.Format("{0:X2}", m.GetIntMask())
        If tbIntrFlag.Text = str Then
            Exit Sub
        Else
            tbIntrFlag.Text = str
        End If
    End Sub
End Class


c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormKeyDisplay↙
    .Designer.vb
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormKeyDisplay
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
```

```
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.Label1 = New System.Windows.Forms.Label()
        Me.tbInput = New System.Windows.Forms.TextBox()
        Me.Label2 = New System.Windows.Forms.Label()
        Me.tbOutput = New System.Windows.Forms.Label()
        Me.SuspendLayout()
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Font = New System.Drawing.Font("Consolas", 14.25!, System.Drawing.FontStyle.Bold,  ↙
    System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label1.ForeColor = System.Drawing.Color.Yellow
        Me.Label1.Location = New System.Drawing.Point(12, 9)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(70, 22)
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "Input:"
        '
        'tbInput
        '
        Me.tbInput.BackColor = System.Drawing.Color.FromArgb(CType(CType(0, Byte), Integer), CType   ↙
    (CType(0, Byte), Integer), CType(CType(192, Byte), Integer))
        Me.tbInput.Font = New System.Drawing.Font("Consolas", 14.25!, System.Drawing.FontStyle.Bold, ↙
    System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.tbInput.ForeColor = System.Drawing.Color.White
        Me.tbInput.Location = New System.Drawing.Point(93, 8)
        Me.tbInput.MaxLength = 25
        Me.tbInput.Name = "tbInput"
        Me.tbInput.Size = New System.Drawing.Size(308, 30)
        Me.tbInput.TabIndex = 1
        '
        'Label2
        '
        Me.Label2.AutoSize = True
        Me.Label2.Font = New System.Drawing.Font("Consolas", 14.25!, System.Drawing.FontStyle.Bold,  ↙
    System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.Label2.ForeColor = System.Drawing.Color.Yellow
        Me.Label2.Location = New System.Drawing.Point(12, 53)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(80, 22)
        Me.Label2.TabIndex = 2
        Me.Label2.Text = "Output:"
        '
        'tbOutput
        '
        Me.tbOutput.AutoSize = True
        Me.tbOutput.Font = New System.Drawing.Font("Consolas", 14.25!, System.Drawing.FontStyle.Bold,↙
     System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.tbOutput.ForeColor = System.Drawing.Color.FromArgb(CType(CType(192, Byte), Integer), CType↙
    (CType(255, Byte), Integer), CType(CType(255, Byte), Integer))
        Me.tbOutput.Location = New System.Drawing.Point(89, 53)
        Me.tbOutput.Name = "tbOutput"
```

```
        Me.tbOutput.Size = New System.Drawing.Size(20, 22)
        Me.tbOutput.TabIndex = 3
        Me.tbOutput.Text = "."
        '
        'FormKeyDisplay
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.BackColor = System.Drawing.Color.FromArgb(CType(CType(0, Byte), Integer), CType(CType(0,
    Byte), Integer), CType(CType(192, Byte), Integer))
        Me.ClientSize = New System.Drawing.Size(430, 118)
        Me.Controls.Add(Me.tbOutput)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.tbInput)
        Me.Controls.Add(Me.Label1)
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog
        Me.MaximizeBox = False
        Me.Name = "FormKeyDisplay"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Keyboard and Display [IN=PORT 00H, OUT=PORT=00H]"
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents tbInput As System.Windows.Forms.TextBox
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents tbOutput As System.Windows.Forms.Label
End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormKeyDisplay
    .vb
********************************************************************************
Imports System.Threading

Public Class FormKeyDisplay

    Private Sub FormKeyDisplay_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
        tbOutput.Text = ""
    End Sub

    Private Sub SerialText(ByVal ch As String)
        FormMain.MakeMeFirst(Me)
        tbOutput.Text += ch
        If tbOutput.Text.Length > 25 Then
            tbOutput.Text = ch
        End If
        If (ch = vbCr) Or (ch = vbCrLf) Then
            tbOutput.Text = ""
        End If
    End Sub

    Public Shared Function PortInWait() As Boolean
        If kd Is Nothing Then
            Return False
        End If
        While kd.tbInput.Text.Length < 1
            Thread.Sleep(50)
        End While
        Return True
    End Function

    Private Function GetAByte() As String
        FormMain.MakeMeFirst(Me)
        Dim str As String
        str = tbInput.Text
        If str.Length < 1 Then
            Return ""
```

```
        Else
            Try
                tbInput.Text = str.Substring(1)
            Catch ex As Exception
                tbInput.Text = ""
            End Try
            Try
                Return str.Substring(0, 1)
            Catch ex As Exception
                Return ""
            End Try
        End If

    End Function

    Private Shared kd As FormKeyDisplay = Nothing
    Public Shared Sub LoadMe(ByVal p As FormMain)
        If kd Is Nothing Then
            kd = New FormKeyDisplay()
            kd.MdiParent = p
            kd.Show()
        End If
    End Sub

    Public Shared Sub OutPort(ByVal p As FormMain, ByVal ch As Integer)
        Try
            LoadMe(p)
            kd.SerialText(Chr(ch And &HFF))
        Catch ex As Exception
            FormReport.WriteLine("Exception in FormKeyDisplay.OutPort(...): " + ex.Message)
        End Try

    End Sub
    Public Shared Function InPort(ByVal p As FormMain) As Integer
        Try
            LoadMe(p)
            Dim str As String
            str = kd.GetAByte()
            If str = "" Then
                Return 0
            Else
                Return Asc(str)
            End If

        Catch ex As Exception

            Return 0
        End Try
    End Function

End Class


c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormKit.Designer.vb
********************************************************************************
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormKit
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub
```

```
    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.Label1 = New System.Windows.Forms.Label()
        Me.Label2 = New System.Windows.Forms.Label()
        Me.tbAddress = New System.Windows.Forms.TextBox()
        Me.tbByte = New System.Windows.Forms.TextBox()
        Me.btnSetAddr = New System.Windows.Forms.Button()
        Me.btnPrev = New System.Windows.Forms.Button()
        Me.btnNext = New System.Windows.Forms.Button()
        Me.btnGo = New System.Windows.Forms.Button()
        Me.btnExec = New System.Windows.Forms.Button()
        Me.btnRes = New System.Windows.Forms.Button()
        Me.SuspendLayout()
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Location = New System.Drawing.Point(12, 21)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(90, 22)
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "Address:"
        '
        'Label2
        '
        Me.Label2.AutoSize = True
        Me.Label2.Location = New System.Drawing.Point(245, 21)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(60, 22)
        Me.Label2.TabIndex = 1
        Me.Label2.Text = "Byte:"
        '
        'tbAddress
        '
        Me.tbAddress.BackColor = System.Drawing.Color.Black
        Me.tbAddress.ForeColor = System.Drawing.Color.Lime
        Me.tbAddress.Location = New System.Drawing.Point(108, 18)
        Me.tbAddress.MaxLength = 4
        Me.tbAddress.Name = "tbAddress"
        Me.tbAddress.ReadOnly = True
        Me.tbAddress.Size = New System.Drawing.Size(100, 30)
        Me.tbAddress.TabIndex = 2
        '
        'tbByte
        '
        Me.tbByte.BackColor = System.Drawing.Color.Black
        Me.tbByte.ForeColor = System.Drawing.Color.Lime
        Me.tbByte.Location = New System.Drawing.Point(311, 18)
        Me.tbByte.MaxLength = 2
        Me.tbByte.Name = "tbByte"
        Me.tbByte.ReadOnly = True
        Me.tbByte.Size = New System.Drawing.Size(64, 30)
        Me.tbByte.TabIndex = 3
        '
        'btnSetAddr
        '
        Me.btnSetAddr.Location = New System.Drawing.Point(16, 68)
        Me.btnSetAddr.Name = "btnSetAddr"
        Me.btnSetAddr.Size = New System.Drawing.Size(146, 38)
        Me.btnSetAddr.TabIndex = 4
        Me.btnSetAddr.Text = "Set Address"
        Me.btnSetAddr.UseVisualStyleBackColor = True
        '
        'btnPrev
```

```
        '
        Me.btnPrev.Location = New System.Drawing.Point(168, 68)
        Me.btnPrev.Name = "btnPrev"
        Me.btnPrev.Size = New System.Drawing.Size(104, 38)
        Me.btnPrev.TabIndex = 5
        Me.btnPrev.Text = "Prev"
        Me.btnPrev.UseVisualStyleBackColor = True
        '
        'btnNext
        '
        Me.btnNext.Location = New System.Drawing.Point(278, 68)
        Me.btnNext.Name = "btnNext"
        Me.btnNext.Size = New System.Drawing.Size(97, 38)
        Me.btnNext.TabIndex = 6
        Me.btnNext.Text = "Next"
        Me.btnNext.UseVisualStyleBackColor = True
        '
        'btnGo
        '
        Me.btnGo.Location = New System.Drawing.Point(16, 112)
        Me.btnGo.Name = "btnGo"
        Me.btnGo.Size = New System.Drawing.Size(146, 38)
        Me.btnGo.TabIndex = 7
        Me.btnGo.Text = "Go"
        Me.btnGo.UseVisualStyleBackColor = True
        '
        'btnExec
        '
        Me.btnExec.Location = New System.Drawing.Point(168, 112)
        Me.btnExec.Name = "btnExec"
        Me.btnExec.Size = New System.Drawing.Size(126, 38)
        Me.btnExec.TabIndex = 8
        Me.btnExec.Text = "Exec"
        Me.btnExec.UseVisualStyleBackColor = True
        '
        'btnRes
        '
        Me.btnRes.Location = New System.Drawing.Point(300, 112)
        Me.btnRes.Name = "btnRes"
        Me.btnRes.Size = New System.Drawing.Size(75, 38)
        Me.btnRes.TabIndex = 9
        Me.btnRes.Text = "Res"
        Me.btnRes.UseVisualStyleBackColor = True
        '
        'FormKit
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(10.0!, 22.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(389, 174)
        Me.Controls.Add(Me.btnRes)
        Me.Controls.Add(Me.btnExec)
        Me.Controls.Add(Me.btnGo)
        Me.Controls.Add(Me.btnNext)
        Me.Controls.Add(Me.btnPrev)
        Me.Controls.Add(Me.btnSetAddr)
        Me.Controls.Add(Me.tbByte)
        Me.Controls.Add(Me.tbAddress)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.Label1)
        Me.Font = New System.Drawing.Font("Consolas", 14.25!, System.Drawing.FontStyle.Bold, System. ↙
    Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog
        Me.Margin = New System.Windows.Forms.Padding(5)
        Me.MaximizeBox = False
        Me.Name = "FormKit"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "8085 Program Kit"
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
```

```
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents tbAddress As System.Windows.Forms.TextBox
    Friend WithEvents tbByte As System.Windows.Forms.TextBox
    Friend WithEvents btnSetAddr As System.Windows.Forms.Button
    Friend WithEvents btnPrev As System.Windows.Forms.Button
    Friend WithEvents btnNext As System.Windows.Forms.Button
    Friend WithEvents btnGo As System.Windows.Forms.Button
    Friend WithEvents btnExec As System.Windows.Forms.Button
    Friend WithEvents btnRes As System.Windows.Forms.Button
End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormKit.vb
*******************************************************************************
Public Class FormKit

    Private Sub btnRes_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↙
    btnRes.Click
        FormMain.GetMachineState().StopExecution()
        tbAddress.Text = "KIT"
        tbByte.Clear()
        tbAddress.ReadOnly = True
        tbByte.ReadOnly = True
    End Sub

    Private Sub FormKit_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↙
    MyBase.Load

        btnRes.PerformClick()
    End Sub

    Private Sub btnSetAddr_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↙
    btnSetAddr.Click
        tbAddress.ReadOnly = False
        tbAddress.Clear()
        tbAddress.Focus()
        tbByte.Clear()
        tbByte.ReadOnly = True
    End Sub

    Private Sub btnNext_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↙
    btnNext.Click
        ShowByte(1)
    End Sub

    Private Sub ShowByte(ByVal prog As Integer)
        Try
            If tbAddress.ReadOnly = True Then
                ' increment address only

                tbByte.ReadOnly = False
                Dim addr As Integer
                addr = Integer.Parse(tbAddress.Text, Globalization.NumberStyles.HexNumber)
                addr = addr And &HFFFF

                Dim bt As Integer
                bt = Integer.Parse(tbByte.Text, Globalization.NumberStyles.HexNumber)
                bt = bt And &HFF

                FormMain.GetMachineState().SetMemory(addr, bt, True)

                addr += prog
                addr = addr And &HFFFF
                tbAddress.Text = String.Format("{0:X4}", addr)
                tbByte.Text = String.Format("{0:X2}", FormMain.GetMachineState().GetMemory(addr))
                tbByte.Focus()
            Else
                ' this is a new address
                tbAddress.ReadOnly = True
```

```
                tbByte.ReadOnly = False

                Dim addr As Integer
                addr = Integer.Parse(tbAddress.Text, Globalization.NumberStyles.HexNumber)
                addr = addr And &HFFFF
                tbAddress.Text = String.Format("{0:X4}", addr)
                tbByte.Text = String.Format("{0:X2}", FormMain.GetMachineState().GetMemory(addr))
                tbByte.Focus()
            End If

        Catch ex As Exception

        End Try


    End Sub

    Private Sub btnPrev_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↙
    btnPrev.Click
        ShowByte(-1)
    End Sub

    Private Sub btnGo_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnGo↙
    .Click
        tbAddress.ReadOnly = False
        tbAddress.Clear()
        tbAddress.Focus()
        tbByte.Clear()
        tbByte.ReadOnly = True
    End Sub

    Private Sub btnExec_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↙
    btnExec.Click
        Try
            FormMain.GetMachineState().StopExecution()

            tbAddress.ReadOnly = True
            tbByte.ReadOnly = True

            Dim addr As Integer
            addr = Integer.Parse(tbAddress.Text, Globalization.NumberStyles.HexNumber)
            addr = addr And &HFFFF
            FormMain.GetMachineState().SetPC(addr, True)
            FormMain.GetMachineState().StartExecution()

            tbAddress.ReadOnly = True
            tbByte.ReadOnly = True

            tbAddress.Clear()
            tbByte.Clear()
        Catch ex As Exception

        End Try
    End Sub

End Class


c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormMain. ↙
    Designer.vb
********************************************************************************
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormMain
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
```

```
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Dim resources As System.ComponentModel.ComponentResourceManager = New System.ComponentModel. ↙
    ComponentResourceManager(GetType(FormMain))
        Me.MenuStrip1 = New System.Windows.Forms.MenuStrip()
        Me.FileToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.NewASMFileToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.OpenASMFileToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.SaveASMFileToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.ExitToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.ViewToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.RegisterStateToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.MemoryViewerToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.PortViewerToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.InterruptToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.MachineStateToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.ResetClearMemoryToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.ResetToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.StartExecutionToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.StopExecutionToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.DeviceToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.ShowAlphanumericKeyboardDisplayToolStripMenuItem = New System.Windows.Forms. ↙
    ToolStripMenuItem()
        Me.HexadecimalDisplayToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.DecimalDisplayToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.BitDisplayToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.TextConsoleDisplayToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.SerialPortToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.ProgrammerToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.AssemblerEditorToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.HexFileDecoderToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.HexFileEncoderToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.HelpToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.AboutToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.Httparnavguddu6tenetToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.IOPortDeviceListToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.KitToolStripMenuItem = New System.Windows.Forms.ToolStripMenuItem()
        Me.MenuStrip1.SuspendLayout()
        Me.SuspendLayout()
        '
        'MenuStrip1
        '
        Me.MenuStrip1.Items.AddRange(New System.Windows.Forms.ToolStripItem() {Me. ↙
    FileToolStripMenuItem, Me.ViewToolStripMenuItem, Me.MachineStateToolStripMenuItem, Me. ↙
    DeviceToolStripMenuItem, Me.ProgrammerToolStripMenuItem, Me.HelpToolStripMenuItem})
        Me.MenuStrip1.Location = New System.Drawing.Point(0, 0)
        Me.MenuStrip1.Name = "MenuStrip1"
        Me.MenuStrip1.Size = New System.Drawing.Size(641, 24)
        Me.MenuStrip1.TabIndex = 1
        Me.MenuStrip1.Text = "MenuStrip1"
        '
        'FileToolStripMenuItem
        '
        Me.FileToolStripMenuItem.DropDownItems.AddRange(New System.Windows.Forms.ToolStripItem() {Me. ↙
    NewASMFileToolStripMenuItem, Me.OpenASMFileToolStripMenuItem, Me.SaveASMFileToolStripMenuItem, Me ↙
    .ExitToolStripMenuItem})
        Me.FileToolStripMenuItem.Name = "FileToolStripMenuItem"
        Me.FileToolStripMenuItem.Size = New System.Drawing.Size(55, 20)
        Me.FileToolStripMenuItem.Text = "Loader"
```

```
        '
        'NewASMFileToolStripMenuItem
        '
        Me.NewASMFileToolStripMenuItem.Name = "NewASMFileToolStripMenuItem"
        Me.NewASMFileToolStripMenuItem.Size = New System.Drawing.Size(169, 22)
        Me.NewASMFileToolStripMenuItem.Text = "Load Program"
        '
        'OpenASMFileToolStripMenuItem
        '
        Me.OpenASMFileToolStripMenuItem.Name = "OpenASMFileToolStripMenuItem"
        Me.OpenASMFileToolStripMenuItem.Size = New System.Drawing.Size(169, 22)
        Me.OpenASMFileToolStripMenuItem.Text = "Load Memory File"
        '
        'SaveASMFileToolStripMenuItem
        '
        Me.SaveASMFileToolStripMenuItem.Name = "SaveASMFileToolStripMenuItem"
        Me.SaveASMFileToolStripMenuItem.Size = New System.Drawing.Size(169, 22)
        Me.SaveASMFileToolStripMenuItem.Text = "Save Memory File"
        '
        'ExitToolStripMenuItem
        '
        Me.ExitToolStripMenuItem.Name = "ExitToolStripMenuItem"
        Me.ExitToolStripMenuItem.Size = New System.Drawing.Size(169, 22)
        Me.ExitToolStripMenuItem.Text = "Exit"
        '
        'ViewToolStripMenuItem
        '
        Me.ViewToolStripMenuItem.DropDownItems.AddRange(New System.Windows.Forms.ToolStripItem() {Me.↙
    RegisterStateToolStripMenuItem, Me.MemoryViewerToolStripMenuItem, Me.PortViewerToolStripMenuItem,↙
     Me.InterruptToolStripMenuItem, Me.KitToolStripMenuItem})
        Me.ViewToolStripMenuItem.Name = "ViewToolStripMenuItem"
        Me.ViewToolStripMenuItem.Size = New System.Drawing.Size(69, 20)
        Me.ViewToolStripMenuItem.Text = "Intel 8085"
        '
        'RegisterStateToolStripMenuItem
        '
        Me.RegisterStateToolStripMenuItem.Name = "RegisterStateToolStripMenuItem"
        Me.RegisterStateToolStripMenuItem.Size = New System.Drawing.Size(189, 22)
        Me.RegisterStateToolStripMenuItem.Text = "Register State"
        '
        'MemoryViewerToolStripMenuItem
        '
        Me.MemoryViewerToolStripMenuItem.Name = "MemoryViewerToolStripMenuItem"
        Me.MemoryViewerToolStripMenuItem.Size = New System.Drawing.Size(189, 22)
        Me.MemoryViewerToolStripMenuItem.Text = "Memory Viewer"
        '
        'PortViewerToolStripMenuItem
        '
        Me.PortViewerToolStripMenuItem.Name = "PortViewerToolStripMenuItem"
        Me.PortViewerToolStripMenuItem.Size = New System.Drawing.Size(189, 22)
        Me.PortViewerToolStripMenuItem.Text = "Port Viewer"
        '
        'InterruptToolStripMenuItem
        '
        Me.InterruptToolStripMenuItem.Name = "InterruptToolStripMenuItem"
        Me.InterruptToolStripMenuItem.Size = New System.Drawing.Size(189, 22)
        Me.InterruptToolStripMenuItem.Text = "Interrupt and Serial IO"
        '
        'MachineStateToolStripMenuItem
        '
        Me.MachineStateToolStripMenuItem.DropDownItems.AddRange(New System.Windows.Forms.            ↙
    ToolStripItem() {Me.ResetClearMemoryToolStripMenuItem, Me.ResetToolStripMenuItem, Me.            ↙
    StartExecutionToolStripMenuItem, Me.StopExecutionToolStripMenuItem})
        Me.MachineStateToolStripMenuItem.Name = "MachineStateToolStripMenuItem"
        Me.MachineStateToolStripMenuItem.Size = New System.Drawing.Size(94, 20)
        Me.MachineStateToolStripMenuItem.Text = "Machine State"
        '
        'ResetClearMemoryToolStripMenuItem
        '
        Me.ResetClearMemoryToolStripMenuItem.Name = "ResetClearMemoryToolStripMenuItem"
        Me.ResetClearMemoryToolStripMenuItem.Size = New System.Drawing.Size(205, 22)
```

```
        Me.ResetClearMemoryToolStripMenuItem.Text = "Reset And Clear Memory"
        '
        'ResetToolStripMenuItem
        '
        Me.ResetToolStripMenuItem.Name = "ResetToolStripMenuItem"
        Me.ResetToolStripMenuItem.Size = New System.Drawing.Size(205, 22)
        Me.ResetToolStripMenuItem.Text = "Reset"
        '
        'StartExecutionToolStripMenuItem
        '
        Me.StartExecutionToolStripMenuItem.Name = "StartExecutionToolStripMenuItem"
        Me.StartExecutionToolStripMenuItem.Size = New System.Drawing.Size(205, 22)
        Me.StartExecutionToolStripMenuItem.Text = "Start Execution"
        '
        'StopExecutionToolStripMenuItem
        '
        Me.StopExecutionToolStripMenuItem.Name = "StopExecutionToolStripMenuItem"
        Me.StopExecutionToolStripMenuItem.Size = New System.Drawing.Size(205, 22)
        Me.StopExecutionToolStripMenuItem.Text = "Stop Execution"
        '
        'DeviceToolStripMenuItem
        '
        Me.DeviceToolStripMenuItem.DropDownItems.AddRange(New System.Windows.Forms.ToolStripItem()
    {Me.ShowAlphanumericKeyboardDisplayToolStripMenuItem, Me.HexadecimalDisplayToolStripMenuItem, Me.
    DecimalDisplayToolStripMenuItem, Me.BitDisplayToolStripMenuItem, Me.
    TextConsoleDisplayToolStripMenuItem, Me.SerialPortToolStripMenuItem})
        Me.DeviceToolStripMenuItem.Name = "DeviceToolStripMenuItem"
        Me.DeviceToolStripMenuItem.Size = New System.Drawing.Size(54, 20)
        Me.DeviceToolStripMenuItem.Text = "Device"
        '
        'ShowAlphanumericKeyboardDisplayToolStripMenuItem
        '
        Me.ShowAlphanumericKeyboardDisplayToolStripMenuItem.Name =
    "ShowAlphanumericKeyboardDisplayToolStripMenuItem"
        Me.ShowAlphanumericKeyboardDisplayToolStripMenuItem.Size = New System.Drawing.Size(298, 22)
        Me.ShowAlphanumericKeyboardDisplayToolStripMenuItem.Text = "Show Alphanumeric Keyboard and
    Display"
        '
        'HexadecimalDisplayToolStripMenuItem
        '
        Me.HexadecimalDisplayToolStripMenuItem.Name = "HexadecimalDisplayToolStripMenuItem"
        Me.HexadecimalDisplayToolStripMenuItem.Size = New System.Drawing.Size(298, 22)
        Me.HexadecimalDisplayToolStripMenuItem.Text = "Hexadecimal Display"
        '
        'DecimalDisplayToolStripMenuItem
        '
        Me.DecimalDisplayToolStripMenuItem.Name = "DecimalDisplayToolStripMenuItem"
        Me.DecimalDisplayToolStripMenuItem.Size = New System.Drawing.Size(298, 22)
        Me.DecimalDisplayToolStripMenuItem.Text = "Decimal Display"
        '
        'BitDisplayToolStripMenuItem
        '
        Me.BitDisplayToolStripMenuItem.Name = "BitDisplayToolStripMenuItem"
        Me.BitDisplayToolStripMenuItem.Size = New System.Drawing.Size(298, 22)
        Me.BitDisplayToolStripMenuItem.Text = "Bit Display"
        '
        'TextConsoleDisplayToolStripMenuItem
        '
        Me.TextConsoleDisplayToolStripMenuItem.Name = "TextConsoleDisplayToolStripMenuItem"
        Me.TextConsoleDisplayToolStripMenuItem.Size = New System.Drawing.Size(298, 22)
        Me.TextConsoleDisplayToolStripMenuItem.Text = "Text and Graphics  Console Display"
        '
        'SerialPortToolStripMenuItem
        '
        Me.SerialPortToolStripMenuItem.Name = "SerialPortToolStripMenuItem"
        Me.SerialPortToolStripMenuItem.Size = New System.Drawing.Size(298, 22)
        Me.SerialPortToolStripMenuItem.Text = "Serial Port"
        '
        'ProgrammerToolStripMenuItem
        '
        Me.ProgrammerToolStripMenuItem.DropDownItems.AddRange(New System.Windows.Forms.ToolStripItem
```

```
    () {Me.AssemblerEditorToolStripMenuItem, Me.HexFileDecoderToolStripMenuItem, Me.
    HexFileEncoderToolStripMenuItem})
        Me.ProgrammerToolStripMenuItem.Name = "ProgrammerToolStripMenuItem"
        Me.ProgrammerToolStripMenuItem.Size = New System.Drawing.Size(86, 20)
        Me.ProgrammerToolStripMenuItem.Text = "Programmer"
        '
        'AssemblerEditorToolStripMenuItem
        '
        Me.AssemblerEditorToolStripMenuItem.Name = "AssemblerEditorToolStripMenuItem"
        Me.AssemblerEditorToolStripMenuItem.Size = New System.Drawing.Size(186, 22)
        Me.AssemblerEditorToolStripMenuItem.Text = "Assembler and Editor"
        '
        'HexFileDecoderToolStripMenuItem
        '
        Me.HexFileDecoderToolStripMenuItem.Name = "HexFileDecoderToolStripMenuItem"
        Me.HexFileDecoderToolStripMenuItem.Size = New System.Drawing.Size(186, 22)
        Me.HexFileDecoderToolStripMenuItem.Text = "Hex Decoder"
        '
        'HexFileEncoderToolStripMenuItem
        '
        Me.HexFileEncoderToolStripMenuItem.Name = "HexFileEncoderToolStripMenuItem"
        Me.HexFileEncoderToolStripMenuItem.Size = New System.Drawing.Size(186, 22)
        Me.HexFileEncoderToolStripMenuItem.Text = "Hex Encoder"
        '
        'HelpToolStripMenuItem
        '
        Me.HelpToolStripMenuItem.DropDownItems.AddRange(New System.Windows.Forms.ToolStripItem() {Me.
    AboutToolStripMenuItem, Me.Httparnavguddu6tenetToolStripMenuItem, Me.
    IOPortDeviceListToolStripMenuItem})
        Me.HelpToolStripMenuItem.Name = "HelpToolStripMenuItem"
        Me.HelpToolStripMenuItem.Size = New System.Drawing.Size(44, 20)
        Me.HelpToolStripMenuItem.Text = "Help"
        '
        'AboutToolStripMenuItem
        '
        Me.AboutToolStripMenuItem.Name = "AboutToolStripMenuItem"
        Me.AboutToolStripMenuItem.Size = New System.Drawing.Size(306, 22)
        Me.AboutToolStripMenuItem.Text = "About"
        '
        'Httparnavguddu6tenetToolStripMenuItem
        '
        Me.Httparnavguddu6tenetToolStripMenuItem.Name = "Httparnavguddu6tenetToolStripMenuItem"
        Me.Httparnavguddu6tenetToolStripMenuItem.Size = New System.Drawing.Size(306, 22)
        Me.Httparnavguddu6tenetToolStripMenuItem.Text = "Homepage URL: http://arnavguddu.6te.net/"
        '
        'IOPortDeviceListToolStripMenuItem
        '
        Me.IOPortDeviceListToolStripMenuItem.Name = "IOPortDeviceListToolStripMenuItem"
        Me.IOPortDeviceListToolStripMenuItem.Size = New System.Drawing.Size(306, 22)
        Me.IOPortDeviceListToolStripMenuItem.Text = "IO Port Device List"
        '
        'KitToolStripMenuItem
        '
        Me.KitToolStripMenuItem.Name = "KitToolStripMenuItem"
        Me.KitToolStripMenuItem.Size = New System.Drawing.Size(189, 22)
        Me.KitToolStripMenuItem.Text = "8085 Kit"
        '
        'FormMain
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(641, 312)
        Me.Controls.Add(Me.MenuStrip1)
        Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
        Me.IsMdiContainer = True
        Me.MainMenuStrip = Me.MenuStrip1
        Me.Name = "FormMain"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Intel 8085 Simulator"
        Me.WindowState = System.Windows.Forms.FormWindowState.Maximized
        Me.MenuStrip1.ResumeLayout(False)
```

```
        Me.MenuStrip1.PerformLayout()
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents MenuStrip1 As System.Windows.Forms.MenuStrip
    Friend WithEvents FileToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents NewASMFileToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents OpenASMFileToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents SaveASMFileToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents ExitToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents ViewToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents RegisterStateToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents MemoryViewerToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents PortViewerToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents InterruptToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents MachineStateToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents ResetClearMemoryToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents ResetToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents StartExecutionToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents StopExecutionToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents HelpToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents AboutToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents ProgrammerToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents AssemblerEditorToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents HexFileDecoderToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents HexFileEncoderToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents HttparnavguddU6tenetToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents IOPortDeviceListToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents DeviceToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents ShowAlphanumericKeyboardDisplayToolStripMenuItem As System.Windows.Forms.
    ToolStripMenuItem
    Friend WithEvents HexadecimalDisplayToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents DecimalDisplayToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents BitDisplayToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents TextConsoleDisplayToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents SerialPortToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents KitToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem

End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormMain.vb
*******************************************************************************
Imports System.IO

Public Class FormMain
    Private Shared machineState As New MachineState()

    Public Shared Function GetMachineState() As MachineState
        Return machineState
    End Function

    Private Sub FormMain_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.
    FormClosingEventArgs) Handles Me.FormClosing
        GetMachineState().StopExecution()
        RemoveHandler GetMachineState().OnStateUpdate, AddressOf OnStateUpdate
        RemoveHandler GetMachineState().OnEventInPortUpdate, AddressOf OnEventInPortUpdate
        RemoveHandler GetMachineState().OnEventOutPortUpdate, AddressOf OnEventOutPortUpdate
    End Sub

    Private Sub FormMain_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    MyBase.Load
        CheckForIllegalCrossThreadCalls = False
        Me.Text += " (ALPHA) "

        FormReport.MainForm = Me
        AddHandler GetMachineState().OnStateUpdate, AddressOf OnStateUpdate
        AddHandler GetMachineState().OnEventInPortUpdate, AddressOf OnEventInPortUpdate
        AddHandler GetMachineState().OnEventOutPortUpdate, AddressOf OnEventOutPortUpdate
```

```
        LoadBios()
    End Sub

    Private Sub ExitToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles ExitToolStripMenuItem.Click
        Me.Dispose()
    End Sub

    Private Sub SaveASMFileToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles SaveASMFileToolStripMenuItem.Click
        Try
            Dim frm As New SaveFileDialog()
            frm.Filter = "Memory Dump File (*.mem) |*.mem|All Files(*.*)|*.*||"
            frm.Title = "Dump Machine and Memory State"

            If frm.ShowDialog() = DialogResult.OK Then
                Dim sw As New StreamWriter(frm.FileName, False)
                Dim dta As Integer
                For i As Integer = 0 To &HFFFF
                    dta = FormMain.GetMachineState.GetMemory(i)
                    sw.Write(String.Format("{0:X2}", dta))
                Next
                sw.Flush()
                sw.Close()
            End If
        Catch ex As Exception

        End Try
    End Sub

    Private Sub OpenASMFileToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles OpenASMFileToolStripMenuItem.Click
        Try
            Dim frm As New OpenFileDialog()
            frm.Filter = "Memory Dump File (*.mem) |*.mem|All Files(*.*)|*.*||"
            frm.Title = "Load Machine and Memory State"

            If frm.ShowDialog() = DialogResult.OK Then
                LoadMemoryFile(frm.FileName)
            End If
        Catch ex As Exception

        End Try
    End Sub

    Private Sub NewASMFileToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles NewASMFileToolStripMenuItem.Click
        Try
            Dim frm As New OpenFileDialog()
            frm.Filter = "Intel Hex File(*.hex)|*.hex|Memory Dump File (*.mem) |*.mem|All Files(*.*)|
    *.*||"
            frm.Title = "Dump Machine and Memory State"

            If frm.ShowDialog() = DialogResult.OK Then
                LoadProgram(frm.FileName)
            End If
        Catch ex As Exception

        End Try
    End Sub

    Private Sub LoadBiosFrom(ByVal fpath As String)
        Try
            Dim pth As New DirectoryInfo(fpath)
            Dim fi() As FileInfo = pth.GetFiles("*.*")
            For Each ff As FileInfo In fi
                LoadProgram(ff.FullName)
            Next
        Catch ex As Exception

        End Try
```

```
    End Sub

    Private Sub LoadBios()
        Try
            Dim bios As New DirectoryInfo(Application.StartupPath + "\\" + "bios")
            If bios.Exists = True Then
                LoadBiosFrom(bios.FullName)
            Else
                bios.Create()
                LoadBios()
            End If
        Catch ex As Exception

        End Try
    End Sub
    Private Sub LoadProgram(ByVal fname As String)
        Try
            Dim fi As New FileInfo(fname)
            If fi.Exists = True Then
                If fi.Extension.ToLower().Trim() = ".mem" Then
                    LoadMemoryFile(fi.FullName)
                ElseIf fi.Extension.ToLower().Trim() = ".hex" Then
                    LoadHexFile(fi.FullName)
                ElseIf fi.Extension.ToLower().Trim() = ".com" Then
                    LoadComFile(fi.FullName)
                ElseIf fi.Extension.ToLower().Trim() = ".bin" Then
                    LoadBinFile(fi.FullName)
                ElseIf fi.Extension.ToLower().Trim() = ".exe" Then
                    LoadExeFile(fi.FullName)
                Else
                    Throw New Exception("Cannot load file " + fi.FullName)
                End If
            End If
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error loading program file", MessageBoxButtons.OK,
    MessageBoxIcon.Stop)
        End Try
    End Sub
    Private Sub LoadBinFile(ByVal fname As String)
        Throw New Exception("Feature not implemented")
    End Sub
    Private Sub LoadComFile(ByVal fname As String)
        Throw New Exception("Feature not implemented")
    End Sub
    Private Sub LoadExeFile(ByVal fname As String)
        Throw New Exception("Feature not implemented")
    End Sub
    Private Sub LoadHexFile(ByVal fname As String)
        Try
            Dim fl As New StreamReader(fname)

            While Not (fl.Peek() = -1)
                Dim chr As Char = ChrW(fl.Read())
                If chr = ":" Then
                    Dim str As String
                    Dim numStyle As Globalization.NumberStyles = Globalization.NumberStyles.HexNumber
                    Dim ifmtprov As IFormatProvider = Globalization.CultureInfo.CurrentCulture

                    Dim ln As Integer = 0
                    str = ChrW(fl.Read)
                    str += ChrW(fl.Read)
                    Integer.TryParse(str, numStyle, ifmtprov, ln)

                    str = ChrW(fl.Read)
                    str += ChrW(fl.Read)
                    str += ChrW(fl.Read)
                    str += ChrW(fl.Read)
                    Dim s_addr As Integer = 0
                    Integer.TryParse(str, numStyle, ifmtprov, s_addr)

                    str = ChrW(fl.Read)
```

```
                    str += ChrW(fl.Read)
                    Dim tt As Integer = 0
                    Integer.TryParse(str, numStyle, ifmtprov, tt)

                    If tt = &H0 Then 'Data
                        While True
                            Dim ch As Integer = fl.Peek
                            If ch = -1 Then
                                Exit While
                            ElseIf ChrW(ch) = ":" Then
                                Exit While
                            End If

                            str = ChrW(fl.Read())
                            str += ChrW(fl.Read())
                            Dim instr As Integer = 0
                            Integer.TryParse(str, numStyle, ifmtprov, instr)

                            ch = fl.Peek()
                            If ch = -1 Or ChrW(ch) = ":" Or ChrW(ch) = vbCr Then
                                Exit While
                            Else
                                GetMachineState().SetMemory(s_addr, instr, True)
                                s_addr += 1
                            End If
                        End While
                    ElseIf tt = &H0 Then 'EOF
                        Exit While
                    End If
                End If
            End While
            fl.Close()
        Catch ex As Exception
            MessageBox.Show(ex.Message.ToString(), "Exception", MessageBoxButtons.OK, MessageBoxIcon.↙
    Stop)
        End Try

    End Sub

    Private Sub LoadMemoryFile(ByVal fname As String)
        Try
            Dim sr As New StreamReader(fname)
            Dim str, str1 As String
            Dim len As Integer
            len = 0
            str = sr.ReadToEnd()
            Dim dta As Integer
            For i As Integer = 0 To &HFFFF
                dta = 0
                Try
                    str1 = str.Substring(len, 2)
                    len += 2
                    dta = Integer.Parse(str1, Globalization.NumberStyles.HexNumber)
                Catch ex1 As Exception

                End Try
                If Not (FormMain.GetMachineState.GetMemory(i) = dta) Then
                    FormMain.GetMachineState.SetMemory(i, dta, True)
                End If
            Next
            sr.Close()
        Catch ex As Exception
            MessageBox.Show(ex.Message.ToString(), "Exception", MessageBoxButtons.OK, MessageBoxIcon.↙
    Stop)
        End Try
    End Sub

    Private Sub RegisterStateToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System↙
    .EventArgs) Handles RegisterStateToolStripMenuItem.Click
        Dim frm As New FormRegisterState()
        frm.MdiParent = Me
```

```
        frm.Show()
    End Sub


    Private Sub ResetClearMemoryToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles ResetClearMemoryToolStripMenuItem.Click
        GetMachineState().StopExecution()
        GetMachineState().ClearMemory()
    End Sub

    Private Sub ResetToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles ResetToolStripMenuItem.Click
        GetMachineState().StopExecution()
        GetMachineState().SetPC(&H0, True)
    End Sub

    Private Sub StartExecutionToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles StartExecutionToolStripMenuItem.Click
        GetMachineState().StartExecution()
    End Sub

    Private Sub StopExecutionToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System
    .EventArgs) Handles StopExecutionToolStripMenuItem.Click
        GetMachineState().StopExecution()
    End Sub

    Private Sub AboutToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles AboutToolStripMenuItem.Click
        MessageBox.Show("Intel 8085 Simulator, Assembler and IDE" + vbCrLf + _
                        "Designed by Arnav Mukhopadhyay", _
                        "Intel 8085 Simulator, Assembler, IDE",
                        MessageBoxButtons.OK, MessageBoxIcon.Information)

    End Sub

    Private Sub MemoryViewerToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles MemoryViewerToolStripMenuItem.Click
        Dim frm As New FormMemoryViewer()
        frm.MdiParent = Me
        frm.Show()

    End Sub

    Public Sub OnStateUpdate(ByVal bRunning As Boolean, ByRef macState As MachineState)
        Dim str As String
        str = Me.Text
        Try
            Dim status As String = ""
            If bRunning = True Then
                status = "RUNNING"
            Else
                status = "STOPPED"
            End If

            If str.IndexOf("[") = -1 Then
                str = String.Format("{0} [{1}]", str, status)
            Else
                str = String.Format("{0}[{1}]", str.Substring(0, str.IndexOf("[")), status)
            End If
        Catch ex As Exception

        End Try

        Me.Text = str
    End Sub

    Private Sub InterruptToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles InterruptToolStripMenuItem.Click
        Dim frm As New FormIntrSerial()
        frm.MdiParent = Me
        frm.Show()
```

```
    End Sub

    Private Sub PortViewerToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles PortViewerToolStripMenuItem.Click
        Dim frm As New FormPortViewer()
        frm.MdiParent = Me
        frm.Show()
    End Sub

    Private Sub HexFileDecoderToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles HexFileDecoderToolStripMenuItem.Click
        Dim frm As New FormHexDecode()
        frm.MdiParent = Me
        frm.Show()
    End Sub

    Private Sub HexFileEncoderToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles HexFileEncoderToolStripMenuItem.Click
        Dim frm As New FormHexEncode()
        frm.MdiParent = Me
        frm.Show()
    End Sub

    Private Sub AssemblerEditorToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles AssemblerEditorToolStripMenuItem.Click
        Dim frm As New FormAssembler()
        frm.MdiParent = Me
        frm.Show()
    End Sub

    Private Sub Httparnavguddu6tenetToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles Httparnavguddu6tenetToolStripMenuItem.Click
        Try
            System.Diagnostics.Process.Start("http://arnavguddu.6te.net/")
        Catch ex As Exception

        End Try
    End Sub

    Private Sub IOPortDeviceListToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles IOPortDeviceListToolStripMenuItem.Click
        Dim sb As String
        sb = "IO Device Port list" + vbCrLf
        sb += "IN PORT = 00H  KEYBOARD INPUT THROUGH TEXT BOX" + vbCrLf
        sb += "OUT PORT = 00H MESSAGE DISPLAY THROUGH TEXT DISPLAY" + vbCrLf
        sb += "IN PORT = 01H  HEX INPUT THROUGH TEXT BOX" + vbCrLf
        sb += "OUT PORT = 01H HEX DISPLAY" + vbCrLf
        sb += "IN PORT = 02H  DECIMAL INPUT THROUGH TEXT BOX" + vbCrLf
        sb += "OUT PORT = 02H DECIMAL DISPLAY" + vbCrLf
        sb += "IN PORT = 03H  BIT INPUT THROUGH BIT DISPLAY" + vbCrLf
        sb += "OUT PORT = 03H BIT OUTPUT THROUGH BIT DISPLAY" + vbCrLf

        sb += "OUT PORT = 0EH COMMAND OUTPUT PORT FOR TEXT AND GRAPHICS CONSOLE DISPLAY " + vbCrLf
        sb += vbTab + "Command bit 0: always 1 to enable" + vbCrLf
        sb += vbTab + "Command bit 1: clear" + vbCrLf
        sb += vbTab + "Command bit 2: 1 for ASCII text display" + vbCrLf
        sb += vbTab + "Command bit 3: 1 for graphics display" + vbCrLf
        sb += vbTab + "Command bit 4: data is x location or row" + vbCrLf
        sb += vbTab + "Command bit 5: data is y location or column" + vbCrLf
        sb += vbTab + "Command bit 6: data is AARRGGBB color" + vbCrLf
        sb += vbTab + "Command bit 7: 1 to repaint" + vbCrLf


        sb += "OUT PORT = 0FH DATA OUTPUT PORT FOR TEXT AND GRAPHICS CONSOLE DISPLAY" + vbCrLf
        sb += "IN PORT = 0FH DATA INPUT PORT FOR TEXT AND GRAPHICS CONSOLE DISPLAY" + vbCrLf
        sb += "IN/OUT 40H = Serial Data IO Port" + vbCrLf
        sb += "OUT 40H = Control Register for Serial Port" + vbCrLf + "IN 40H = Status Register for
    Serial Port" + vbCrLf

        MessageBox.Show(sb, "IO Device Port List", MessageBoxButtons.OK, MessageBoxIcon.Information)
```

```
    End Sub


    Private Delegate Sub dOutPort(ByVal p As FormMain, ByVal data As Integer)
    Private Delegate Function dInPort(ByVal p As FormMain) As Integer
    Public Sub OnEventOutPortUpdate(ByVal data As Byte, ByVal port As Byte, ByRef macState As MachineState)
        Try
            If port = 0 Then
                Invoke(New dOutPort(AddressOf FormKeyDisplay.OutPort), New Object() {Me, data})
            ElseIf port = 1 Then
                Invoke(New dOutPort(AddressOf FormHexIO.OutPort), New Object() {Me, data})
            ElseIf port = 2 Then
                Invoke(New dOutPort(AddressOf FormDecDisplay.OutPort), New Object() {Me, data})
            ElseIf port = 3 Then
                Invoke(New dOutPort(AddressOf FormBitDisplay.OutPort), New Object() {Me, data})
            ElseIf port = &HE Then
                Invoke(New dOutPort(AddressOf FormScreenDis.OutCommandPort), New Object() {Me, data})
            ElseIf port = &HF Then
                Invoke(New dOutPort(AddressOf FormScreenDis.OutPort), New Object() {Me, data})
            ElseIf port = &H40 Then
                If FormUsart.PortInWait() = False Then
                    Invoke(New dLoadMe(AddressOf FormUsart.LoadMe), New Object() {Me})
                    FormDecDisplay.PortInWait()
                End If
                Invoke(New dOutPort(AddressOf FormUsart.OutPort), New Object() {Me, data})
            ElseIf port = &H41 Then
                If FormUsart.PortInWait() = False Then
                    Invoke(New dLoadMe(AddressOf FormUsart.LoadMe), New Object() {Me})
                    FormDecDisplay.PortInWait()
                End If
                Invoke(New dOutPort(AddressOf FormUsart.OutPortCommand), New Object() {Me, data})
            End If
        Catch ex As Exception

        End Try

    End Sub
    Private Delegate Sub dLoadMe(ByVal frm As FormMain)
    Public Sub OnEventInPortUpdate(ByRef data As Byte, ByVal port As Byte, ByRef macState As MachineState, ByRef prio As Integer) ' higher prio will be able to push in data

        Dim bHandled As Boolean
        bHandled = False
        Try
            If port = 0 Then
                bHandled = True
                If FormKeyDisplay.PortInWait() = False Then
                    Invoke(New dLoadMe(AddressOf FormKeyDisplay.LoadMe), New Object() {Me})
                    FormKeyDisplay.PortInWait()
                End If
                data = Invoke(New dInPort(AddressOf FormKeyDisplay.InPort), New Object() {Me})
            ElseIf port = 1 Then
                bHandled = True
                If FormHexIO.PortInWait() = False Then
                    Invoke(New dLoadMe(AddressOf FormHexIO.LoadMe), New Object() {Me})
                    FormHexIO.PortInWait()
                End If
                data = Invoke(New dInPort(AddressOf FormHexIO.InPort), New Object() {Me})
            ElseIf port = 2 Then
                bHandled = True
                If FormDecDisplay.PortInWait() = False Then
                    Invoke(New dLoadMe(AddressOf FormDecDisplay.LoadMe), New Object() {Me})
                    FormDecDisplay.PortInWait()
                End If
                data = Invoke(New dInPort(AddressOf FormDecDisplay.InPort), New Object() {Me})
            ElseIf port = 3 Then
                bHandled = True
                data = Invoke(New dInPort(AddressOf FormBitDisplay.InPort), New Object() {Me})
            ElseIf port = &H40 Then
                bHandled = True
```

```
                If FormUsart.PortInWait() = False Then
                    Invoke(New dLoadMe(AddressOf FormUsart.LoadMe), New Object() {Me})
                    FormDecDisplay.PortInWait()
                End If
                data = Invoke(New dInPort(AddressOf FormUsart.InPort), New Object() {Me})
            ElseIf port = &H41 Then
                bHandled = True
                If FormUsart.PortInWait() = False Then
                    Invoke(New dLoadMe(AddressOf FormUsart.LoadMe), New Object() {Me})
                    FormDecDisplay.PortInWait()
                End If
                data = Invoke(New dInPort(AddressOf FormUsart.InPortCommand), New Object() {Me})
            End If
        Catch ex As Exception
            bHandled = False

        End Try

        If bHandled = True Then
            prio = 10000
        End If
    End Sub

    Private Sub ShowAlphanumericKeyboardDisplayToolStripMenuItem_Click(ByVal sender As System.Object, ↙
     ByVal e As System.EventArgs) Handles ShowAlphanumericKeyboardDisplayToolStripMenuItem.Click
        FormKeyDisplay.LoadMe(Me)
    End Sub

    Private Sub HexadecimalDisplayToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As  ↙
    System.EventArgs) Handles HexadecimalDisplayToolStripMenuItem.Click
        FormHexIO.LoadMe(Me)
    End Sub

    Private Sub DecimalDisplayToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As      ↙
    System.EventArgs) Handles DecimalDisplayToolStripMenuItem.Click
        FormDecDisplay.LoadMe(Me)
    End Sub

    Private Sub BitDisplayToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.  ↙
    EventArgs) Handles BitDisplayToolStripMenuItem.Click
        FormBitDisplay.LoadMe(Me)
    End Sub

    Private Sub TextConsoleDisplayToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As  ↙
    System.EventArgs) Handles TextConsoleDisplayToolStripMenuItem.Click
        FormScreenDis.LoadMe(Me)
    End Sub

    Public Shared Sub MakeMeFirst(ByVal frm As Form)
        If frm Is Nothing Then
            Exit Sub
        End If

        frm.BringToFront()
    End Sub

    Private Sub SerialPortToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.  ↙
    EventArgs) Handles SerialPortToolStripMenuItem.Click
        FormUsart.LoadMe(Me)
    End Sub

    Private Sub KitToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.         ↙
    EventArgs) Handles KitToolStripMenuItem.Click
        Dim frm As New FormKit()
        frm.MdiParent = Me
        frm.Show()
    End Sub
End Class
```

```
c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\
    FormMemoryViewer.Designer.vb
*******************************************************************************
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormMemoryViewer
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.dgview = New System.Windows.Forms.DataGridView()
        CType(Me.dgview, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.SuspendLayout()
        '
        'dgview
        '
        Me.dgview.AllowUserToAddRows = False
        Me.dgview.AllowUserToDeleteRows = False
        Me.dgview.AllowUserToResizeColumns = False
        Me.dgview.AllowUserToResizeRows = False
        Me.dgview.AutoSizeColumnsMode = System.Windows.Forms.DataGridViewAutoSizeColumnsMode.
    DisplayedCells
        Me.dgview.AutoSizeRowsMode = System.Windows.Forms.DataGridViewAutoSizeRowsMode.DisplayedCells
        Me.dgview.BorderStyle = System.Windows.Forms.BorderStyle.None
        Me.dgview.CausesValidation = False
        Me.dgview.ClipboardCopyMode = System.Windows.Forms.DataGridViewClipboardCopyMode.
    EnableWithoutHeaderText
        Me.dgview.ColumnHeadersHeightSizeMode = System.Windows.Forms.
    DataGridViewColumnHeadersHeightSizeMode.AutoSize
        Me.dgview.Dock = System.Windows.Forms.DockStyle.Fill
        Me.dgview.Location = New System.Drawing.Point(0, 0)
        Me.dgview.MultiSelect = False
        Me.dgview.Name = "dgview"
        Me.dgview.RowHeadersWidth = 60
        Me.dgview.SelectionMode = System.Windows.Forms.DataGridViewSelectionMode.CellSelect
        Me.dgview.ShowCellToolTips = False
        Me.dgview.ShowEditingIcon = False
        Me.dgview.Size = New System.Drawing.Size(699, 460)
        Me.dgview.TabIndex = 0
        Me.dgview.TabStop = False
        '
        'FormMemoryViewer
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(699, 460)
        Me.Controls.Add(Me.dgview)
        Me.Name = "FormMemoryViewer"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Memory Viewer"
        CType(Me.dgview, System.ComponentModel.ISupportInitialize).EndInit()
        Me.ResumeLayout(False)

    End Sub
```

```
    Friend WithEvents dgview As System.Windows.Forms.DataGridView
End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\
    FormMemoryViewer.vb
********************************************************************************
Public Class FormMemoryViewer
    Private Shared dt As DataTable = Nothing

    Private Sub FormMemoryViewer_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.
    FormClosingEventArgs) Handles Me.FormClosing
        RemoveHandler FormMain.GetMachineState().OnEventMemoryUpdate, AddressOf OnEventMemoryUpdate

    End Sub

    Private Sub FormMemoryViewer_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
        LoadMemory()
        AddHandler FormMain.GetMachineState().OnEventMemoryUpdate, AddressOf OnEventMemoryUpdate

    End Sub
    ' Address : Row(Hi):Col(Lo)


    Public Sub LoadMemory()
        Try
            If dt Is Nothing Then

                dt = New DataTable()
                Dim dc As DataColumn
                Dim dr As DataRow

                Dim str, str1 As String

                For col As Integer = 0 To &HFF
                    str = String.Format("{0:X2}", col)
                    dc = New DataColumn(str, GetType(String))
                    dt.Columns.Add(dc)
                Next

                For row As Integer = 0 To &HFF
                    dr = dt.NewRow()
                    For col As Integer = 0 To &HFF
                        str = String.Format("{0:X2}", col)
                        str1 = String.Format("{0:X2}", FormMain.GetMachineState().GetMemory((row <<
    8) Or col))
                        dr(str) = str1
                    Next

                    dt.Rows.Add(dr)
                Next
            End If

            dgview.DataSource = dt
            dgview.RowHeadersVisible = True

        Catch ex As Exception
            MessageBox.Show(ex.ToString(), "Exception", MessageBoxButtons.OK, MessageBoxIcon.Stop)
        End Try
    End Sub
    Private Sub RefreshDGView()
        dgview.Width = Me.Width
        dgview.Height = Me.Height
        dgview.Left = 0
        dgview.Top = 0
    End Sub

    Private Sub dgview_CellEndEdit(ByVal sender As Object, ByVal e As System.Windows.Forms.
    DataGridViewCellEventArgs) Handles dgview.CellEndEdit
```

```
        Dim row, col, addr As Integer
        row = e.RowIndex
        col = e.ColumnIndex
        addr = (row << 8) Or col

        Dim str As String
        str = dgview.Rows(row).Cells.Item(col).Value.ToString()

        Dim data As Integer = 0
        If Integer.TryParse(str, System.Globalization.NumberStyles.HexNumber, Nothing, data) = True ↙
    Then
            FormMain.GetMachineState().SetMemory(addr, data And &HFF, False)
            Dim valx As Integer
            valx = FormMain.GetMachineState().GetMemory(addr)
            If Not (valx = data) Then
                dgview.Rows(row).Cells.Item(col).Value = valx.ToString()
            End If
        End If
    End Sub

    Private Sub dgview_DataBindingComplete(ByVal sender As Object, ByVal e As System.Windows.Forms. ↙
    DataGridViewBindingCompleteEventArgs) Handles dgview.DataBindingComplete
        If dgview.Rows(0).HeaderCell.Value Is Nothing Then
        ElseIf dgview.Rows(0).HeaderCell.Value.ToString().Trim.Length < 1 Then
        Else
            Return
        End If

        Dim str As String
        For row As Integer = 0 To &HFF
            str = String.Format("{0:X2}", row)
            dgview.Rows(row).HeaderCell.Value = str
        Next
    End Sub

    Public Sub OnEventMemoryUpdate(ByVal addr As UInt16, ByRef macState As MachineState)
        Dim row, col As Integer
        Dim data As Byte = macState.GetMemory(addr)
        col = addr And &HFF
        row = (addr >> 8) And &HFF

        Dim str As String
        str = String.Format("{0:X2}", data)
        dgview.Rows(row).Cells.Item(col).Value = str
    End Sub

End Class


c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormPortViewer ↙
    .Designer.vb
********************************************************************************
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormPortViewer
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer
```

```
    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.dgvw = New System.Windows.Forms.DataGridView()
        CType(Me.dgvw, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.SuspendLayout()
        '
        'dgvw
        '
        Me.dgvw.AllowUserToAddRows = False
        Me.dgvw.AllowUserToDeleteRows = False
        Me.dgvw.AllowUserToResizeColumns = False
        Me.dgvw.AllowUserToResizeRows = False
        Me.dgvw.AutoSizeColumnsMode = System.Windows.Forms.DataGridViewAutoSizeColumnsMode.Fill
        Me.dgvw.ColumnHeadersHeightSizeMode = System.Windows.Forms.
    DataGridViewColumnHeadersHeightSizeMode.AutoSize
        Me.dgvw.Dock = System.Windows.Forms.DockStyle.Fill
        Me.dgvw.Location = New System.Drawing.Point(0, 0)
        Me.dgvw.MultiSelect = False
        Me.dgvw.Name = "dgvw"
        Me.dgvw.Size = New System.Drawing.Size(679, 382)
        Me.dgvw.TabIndex = 0
        '
        'FormPortViewer
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(679, 382)
        Me.Controls.Add(Me.dgvw)
        Me.Name = "FormPortViewer"
        Me.Text = "FormPortViewer"
        CType(Me.dgvw, System.ComponentModel.ISupportInitialize).EndInit()
        Me.ResumeLayout(False)

    End Sub
    Friend WithEvents dgvw As System.Windows.Forms.DataGridView
End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormPortViewer
    .vb
********************************************************************************
Public Class FormPortViewer
    Public Shared PortPriority As Integer = 1
    Private Shared dt As DataTable = Nothing

    Private Sub FormPortViewer_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.
    FormClosingEventArgs) Handles Me.FormClosing
        RemoveHandler FormMain.GetMachineState.OnEventInPortUpdate, AddressOf OnEventInPortUpdate
        RemoveHandler FormMain.GetMachineState.OnEventOutPortUpdate, AddressOf OnEventOutPortUpdate

    End Sub

    Private Sub FormPortViewer_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
        If dt Is Nothing Then
            dt = New DataTable()
            Dim dr As DataRow
            Dim dc As DataColumn
            Dim str As String

            For col As Integer = 0 To &HF
                str = String.Format("{0:X}", col)
                dc = New DataColumn(str, GetType(String))
                dt.Columns.Add(dc)
            Next
            For row As Integer = 0 To &HF
                dr = dt.NewRow()
```

```
                For col As Integer = 0 To &HF
                    str = String.Format("{0:X}", col)
                    dr(str) = "0"
                Next
                dt.Rows.Add(dr)
            Next
        End If

        dgvw.DataSource = dt
        dgvw.RowHeadersVisible = True

        AddHandler FormMain.GetMachineState.OnEventInPortUpdate, AddressOf OnEventInPortUpdate
        AddHandler FormMain.GetMachineState.OnEventOutPortUpdate, AddressOf OnEventOutPortUpdate
    End Sub

    Private Sub dgvw_DataBindingComplete(ByVal sender As Object, ByVal e As System.Windows.Forms.
    DataGridViewBindingCompleteEventArgs) Handles dgvw.DataBindingComplete
        If dgvw.Rows(0).HeaderCell.Value Is Nothing Then
        ElseIf dgvw.Rows(0).HeaderCell.Value.ToString().Trim.Length < 1 Then
        Else
            Return
        End If
        Dim str As String

        For row As Integer = 0 To &HF
            str = String.Format("{0:X}", row)
            dgvw.Rows(row).HeaderCell.Value = str
        Next

    End Sub

    Public Sub OnEventOutPortUpdate(ByVal data As Byte, ByVal port As Byte, ByRef macState As
    MachineState)
        Dim row, col As Integer
        col = port And &HF
        row = (port >> 4) And &HF
        dgvw.Rows(row).Cells.Item(col).Value = String.Format("{0:X2}", data)
    End Sub
    Public Sub OnEventInPortUpdate(ByRef data As Byte, ByVal port As Byte, ByRef macState As
    MachineState, ByRef prio As Integer)
        Dim row, col As Integer
        col = port And &HF
        row = (port >> 4) And &HF
        Dim str As String
        str = dgvw.Rows(row).Cells.Item(col).Value
        Dim val As Integer = 0
        Try
            val = Integer.Parse(str, Globalization.NumberStyles.HexNumber)
        Catch ex As Exception

        End Try
        val = val And &HFF
        If FormPortViewer.PortPriority > prio Then
            ' push out data
            data = val
        Else
            OnEventOutPortUpdate(data, port, macState)
        End If
    End Sub

End Class


c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\
    FormRegisterState.Designer.vb
*******************************************************************************
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormRegisterState
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
```

```
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.GroupBox1 = New System.Windows.Forms.GroupBox()
        Me.tbS = New System.Windows.Forms.TextBox()
        Me.Label6 = New System.Windows.Forms.Label()
        Me.tbZ = New System.Windows.Forms.TextBox()
        Me.Label5 = New System.Windows.Forms.Label()
        Me.tbAC = New System.Windows.Forms.TextBox()
        Me.Label4 = New System.Windows.Forms.Label()
        Me.tbP = New System.Windows.Forms.TextBox()
        Me.Label3 = New System.Windows.Forms.Label()
        Me.tbCY = New System.Windows.Forms.TextBox()
        Me.Label2 = New System.Windows.Forms.Label()
        Me.tbFlag = New System.Windows.Forms.TextBox()
        Me.Label1 = New System.Windows.Forms.Label()
        Me.GroupBox2 = New System.Windows.Forms.GroupBox()
        Me.tbM = New System.Windows.Forms.TextBox()
        Me.Label14 = New System.Windows.Forms.Label()
        Me.tbL = New System.Windows.Forms.TextBox()
        Me.Label13 = New System.Windows.Forms.Label()
        Me.tbH = New System.Windows.Forms.TextBox()
        Me.Label12 = New System.Windows.Forms.Label()
        Me.tbE = New System.Windows.Forms.TextBox()
        Me.Label11 = New System.Windows.Forms.Label()
        Me.tbD = New System.Windows.Forms.TextBox()
        Me.Label10 = New System.Windows.Forms.Label()
        Me.tbC = New System.Windows.Forms.TextBox()
        Me.Label9 = New System.Windows.Forms.Label()
        Me.tbB = New System.Windows.Forms.TextBox()
        Me.Label8 = New System.Windows.Forms.Label()
        Me.tbA = New System.Windows.Forms.TextBox()
        Me.Label7 = New System.Windows.Forms.Label()
        Me.GroupBox3 = New System.Windows.Forms.GroupBox()
        Me.tbBC = New System.Windows.Forms.TextBox()
        Me.Label15 = New System.Windows.Forms.Label()
        Me.tbDE = New System.Windows.Forms.TextBox()
        Me.Label16 = New System.Windows.Forms.Label()
        Me.tbHL = New System.Windows.Forms.TextBox()
        Me.Label17 = New System.Windows.Forms.Label()
        Me.tbSP = New System.Windows.Forms.TextBox()
        Me.Label18 = New System.Windows.Forms.Label()
        Me.GroupBox4 = New System.Windows.Forms.GroupBox()
        Me.tbPC = New System.Windows.Forms.TextBox()
        Me.Label19 = New System.Windows.Forms.Label()
        Me.tbIR = New System.Windows.Forms.TextBox()
        Me.Label20 = New System.Windows.Forms.Label()
        Me.tbDiasm = New System.Windows.Forms.TextBox()
        Me.GroupBox5 = New System.Windows.Forms.GroupBox()
        Me.tbClock = New System.Windows.Forms.Label()
        Me.lbOriginal = New System.Windows.Forms.Label()
        Me.tckBarClock = New System.Windows.Forms.TrackBar()
        Me.GroupBox1.SuspendLayout()
        Me.GroupBox2.SuspendLayout()
        Me.GroupBox3.SuspendLayout()
```

```
        Me.GroupBox4.SuspendLayout()
        Me.GroupBox5.SuspendLayout()
        CType(Me.tckBarClock, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.SuspendLayout()
        '
        'GroupBox1
        '
        Me.GroupBox1.Controls.Add(Me.tbS)
        Me.GroupBox1.Controls.Add(Me.Label6)
        Me.GroupBox1.Controls.Add(Me.tbZ)
        Me.GroupBox1.Controls.Add(Me.Label5)
        Me.GroupBox1.Controls.Add(Me.tbAC)
        Me.GroupBox1.Controls.Add(Me.Label4)
        Me.GroupBox1.Controls.Add(Me.tbP)
        Me.GroupBox1.Controls.Add(Me.Label3)
        Me.GroupBox1.Controls.Add(Me.tbCY)
        Me.GroupBox1.Controls.Add(Me.Label2)
        Me.GroupBox1.Controls.Add(Me.tbFlag)
        Me.GroupBox1.Controls.Add(Me.Label1)
        Me.GroupBox1.Location = New System.Drawing.Point(12, 12)
        Me.GroupBox1.Name = "GroupBox1"
        Me.GroupBox1.Size = New System.Drawing.Size(451, 51)
        Me.GroupBox1.TabIndex = 0
        Me.GroupBox1.TabStop = False
        Me.GroupBox1.Text = "Flag Register:"
        '
        'tbS
        '
        Me.tbS.Location = New System.Drawing.Point(394, 19)
        Me.tbS.Name = "tbS"
        Me.tbS.ReadOnly = True
        Me.tbS.Size = New System.Drawing.Size(32, 20)
        Me.tbS.TabIndex = 11
        Me.tbS.TabStop = False
        '
        'Label6
        '
        Me.Label6.AutoSize = True
        Me.Label6.Location = New System.Drawing.Point(371, 22)
        Me.Label6.Name = "Label6"
        Me.Label6.Size = New System.Drawing.Size(17, 13)
        Me.Label6.TabIndex = 10
        Me.Label6.Text = "S:"
        '
        'tbZ
        '
        Me.tbZ.Location = New System.Drawing.Point(334, 19)
        Me.tbZ.Name = "tbZ"
        Me.tbZ.ReadOnly = True
        Me.tbZ.Size = New System.Drawing.Size(32, 20)
        Me.tbZ.TabIndex = 9
        Me.tbZ.TabStop = False
        '
        'Label5
        '
        Me.Label5.AutoSize = True
        Me.Label5.Location = New System.Drawing.Point(311, 22)
        Me.Label5.Name = "Label5"
        Me.Label5.Size = New System.Drawing.Size(17, 13)
        Me.Label5.TabIndex = 8
        Me.Label5.Text = "Z:"
        '
        'tbAC
        '
        Me.tbAC.Location = New System.Drawing.Point(275, 19)
        Me.tbAC.Name = "tbAC"
        Me.tbAC.ReadOnly = True
        Me.tbAC.Size = New System.Drawing.Size(30, 20)
        Me.tbAC.TabIndex = 7
        Me.tbAC.TabStop = False
        '
```

```
        'Label4
        '
        Me.Label4.AutoSize = True
        Me.Label4.Location = New System.Drawing.Point(245, 22)
        Me.Label4.Name = "Label4"
        Me.Label4.Size = New System.Drawing.Size(24, 13)
        Me.Label4.TabIndex = 6
        Me.Label4.Text = "AC:"
        '
        'tbP
        '
        Me.tbP.Location = New System.Drawing.Point(210, 19)
        Me.tbP.Name = "tbP"
        Me.tbP.ReadOnly = True
        Me.tbP.Size = New System.Drawing.Size(29, 20)
        Me.tbP.TabIndex = 5
        Me.tbP.TabStop = False
        '
        'Label3
        '
        Me.Label3.AutoSize = True
        Me.Label3.Location = New System.Drawing.Point(187, 22)
        Me.Label3.Name = "Label3"
        Me.Label3.Size = New System.Drawing.Size(17, 13)
        Me.Label3.TabIndex = 4
        Me.Label3.Text = "P:"
        '
        'tbCY
        '
        Me.tbCY.Location = New System.Drawing.Point(146, 19)
        Me.tbCY.Name = "tbCY"
        Me.tbCY.ReadOnly = True
        Me.tbCY.Size = New System.Drawing.Size(35, 20)
        Me.tbCY.TabIndex = 3
        Me.tbCY.TabStop = False
        '
        'Label2
        '
        Me.Label2.AutoSize = True
        Me.Label2.Location = New System.Drawing.Point(119, 22)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(24, 13)
        Me.Label2.TabIndex = 2
        Me.Label2.Text = "CY:"
        '
        'tbFlag
        '
        Me.tbFlag.Location = New System.Drawing.Point(50, 19)
        Me.tbFlag.Name = "tbFlag"
        Me.tbFlag.ReadOnly = True
        Me.tbFlag.Size = New System.Drawing.Size(59, 20)
        Me.tbFlag.TabIndex = 1
        Me.tbFlag.TabStop = False
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Location = New System.Drawing.Point(14, 22)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(30, 13)
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "Flag:"
        '
        'GroupBox2
        '
        Me.GroupBox2.Controls.Add(Me.tbM)
        Me.GroupBox2.Controls.Add(Me.Label14)
        Me.GroupBox2.Controls.Add(Me.tbL)
        Me.GroupBox2.Controls.Add(Me.Label13)
        Me.GroupBox2.Controls.Add(Me.tbH)
        Me.GroupBox2.Controls.Add(Me.Label12)
```

```
        Me.GroupBox2.Controls.Add(Me.tbE)
        Me.GroupBox2.Controls.Add(Me.Label11)
        Me.GroupBox2.Controls.Add(Me.tbD)
        Me.GroupBox2.Controls.Add(Me.Label10)
        Me.GroupBox2.Controls.Add(Me.tbC)
        Me.GroupBox2.Controls.Add(Me.Label9)
        Me.GroupBox2.Controls.Add(Me.tbB)
        Me.GroupBox2.Controls.Add(Me.Label8)
        Me.GroupBox2.Controls.Add(Me.tbA)
        Me.GroupBox2.Controls.Add(Me.Label7)
        Me.GroupBox2.Location = New System.Drawing.Point(16, 73)
        Me.GroupBox2.Name = "GroupBox2"
        Me.GroupBox2.Size = New System.Drawing.Size(446, 109)
        Me.GroupBox2.TabIndex = 1
        Me.GroupBox2.TabStop = False
        Me.GroupBox2.Text = "8-Bit Registers:"
        '
        'tbM
        '
        Me.tbM.Location = New System.Drawing.Point(46, 41)
        Me.tbM.Name = "tbM"
        Me.tbM.ReadOnly = True
        Me.tbM.Size = New System.Drawing.Size(59, 20)
        Me.tbM.TabIndex = 17
        Me.tbM.TabStop = False
        '
        'Label14
        '
        Me.Label14.AutoSize = True
        Me.Label14.Location = New System.Drawing.Point(10, 44)
        Me.Label14.Name = "Label14"
        Me.Label14.Size = New System.Drawing.Size(19, 13)
        Me.Label14.TabIndex = 16
        Me.Label14.Text = "M:"
        '
        'tbL
        '
        Me.tbL.Location = New System.Drawing.Point(310, 71)
        Me.tbL.Name = "tbL"
        Me.tbL.ReadOnly = True
        Me.tbL.Size = New System.Drawing.Size(59, 20)
        Me.tbL.TabIndex = 15
        Me.tbL.TabStop = False
        '
        'Label13
        '
        Me.Label13.AutoSize = True
        Me.Label13.Location = New System.Drawing.Point(274, 74)
        Me.Label13.Name = "Label13"
        Me.Label13.Size = New System.Drawing.Size(16, 13)
        Me.Label13.TabIndex = 14
        Me.Label13.Text = "L:"
        '
        'tbH
        '
        Me.tbH.Location = New System.Drawing.Point(186, 71)
        Me.tbH.Name = "tbH"
        Me.tbH.ReadOnly = True
        Me.tbH.Size = New System.Drawing.Size(59, 20)
        Me.tbH.TabIndex = 13
        Me.tbH.TabStop = False
        '
        'Label12
        '
        Me.Label12.AutoSize = True
        Me.Label12.Location = New System.Drawing.Point(150, 74)
        Me.Label12.Name = "Label12"
        Me.Label12.Size = New System.Drawing.Size(18, 13)
        Me.Label12.TabIndex = 12
        Me.Label12.Text = "H:"
        '
```

```
        'tbE
        '
        Me.tbE.Location = New System.Drawing.Point(310, 45)
        Me.tbE.Name = "tbE"
        Me.tbE.ReadOnly = True
        Me.tbE.Size = New System.Drawing.Size(59, 20)
        Me.tbE.TabIndex = 11
        Me.tbE.TabStop = False
        '
        'Label11
        '
        Me.Label11.AutoSize = True
        Me.Label11.Location = New System.Drawing.Point(274, 48)
        Me.Label11.Name = "Label11"
        Me.Label11.Size = New System.Drawing.Size(17, 13)
        Me.Label11.TabIndex = 10
        Me.Label11.Text = "E:"
        '
        'tbD
        '
        Me.tbD.Location = New System.Drawing.Point(186, 45)
        Me.tbD.Name = "tbD"
        Me.tbD.ReadOnly = True
        Me.tbD.Size = New System.Drawing.Size(59, 20)
        Me.tbD.TabIndex = 9
        Me.tbD.TabStop = False
        '
        'Label10
        '
        Me.Label10.AutoSize = True
        Me.Label10.Location = New System.Drawing.Point(150, 48)
        Me.Label10.Name = "Label10"
        Me.Label10.Size = New System.Drawing.Size(18, 13)
        Me.Label10.TabIndex = 8
        Me.Label10.Text = "D:"
        '
        'tbC
        '
        Me.tbC.Location = New System.Drawing.Point(310, 19)
        Me.tbC.Name = "tbC"
        Me.tbC.ReadOnly = True
        Me.tbC.Size = New System.Drawing.Size(59, 20)
        Me.tbC.TabIndex = 7
        Me.tbC.TabStop = False
        '
        'Label9
        '
        Me.Label9.AutoSize = True
        Me.Label9.Location = New System.Drawing.Point(274, 22)
        Me.Label9.Name = "Label9"
        Me.Label9.Size = New System.Drawing.Size(17, 13)
        Me.Label9.TabIndex = 6
        Me.Label9.Text = "C:"
        '
        'tbB
        '
        Me.tbB.Location = New System.Drawing.Point(186, 19)
        Me.tbB.Name = "tbB"
        Me.tbB.ReadOnly = True
        Me.tbB.Size = New System.Drawing.Size(59, 20)
        Me.tbB.TabIndex = 5
        Me.tbB.TabStop = False
        '
        'Label8
        '
        Me.Label8.AutoSize = True
        Me.Label8.Location = New System.Drawing.Point(150, 22)
        Me.Label8.Name = "Label8"
        Me.Label8.Size = New System.Drawing.Size(17, 13)
        Me.Label8.TabIndex = 4
        Me.Label8.Text = "B:"
```

```
        '
        'tbA
        '
        Me.tbA.Location = New System.Drawing.Point(46, 19)
        Me.tbA.Name = "tbA"
        Me.tbA.ReadOnly = True
        Me.tbA.Size = New System.Drawing.Size(59, 20)
        Me.tbA.TabIndex = 3
        Me.tbA.TabStop = False
        '
        'Label7
        '
        Me.Label7.AutoSize = True
        Me.Label7.Location = New System.Drawing.Point(10, 22)
        Me.Label7.Name = "Label7"
        Me.Label7.Size = New System.Drawing.Size(17, 13)
        Me.Label7.TabIndex = 2
        Me.Label7.Text = "A:"
        '
        'GroupBox3
        '
        Me.GroupBox3.Controls.Add(Me.tbSP)
        Me.GroupBox3.Controls.Add(Me.Label18)
        Me.GroupBox3.Controls.Add(Me.tbHL)
        Me.GroupBox3.Controls.Add(Me.Label17)
        Me.GroupBox3.Controls.Add(Me.tbDE)
        Me.GroupBox3.Controls.Add(Me.Label16)
        Me.GroupBox3.Controls.Add(Me.tbBC)
        Me.GroupBox3.Controls.Add(Me.Label15)
        Me.GroupBox3.Location = New System.Drawing.Point(18, 203)
        Me.GroupBox3.Name = "GroupBox3"
        Me.GroupBox3.Size = New System.Drawing.Size(443, 62)
        Me.GroupBox3.TabIndex = 2
        Me.GroupBox3.TabStop = False
        Me.GroupBox3.Text = "16 Bit Registers:"
        '
        'tbBC
        '
        Me.tbBC.Location = New System.Drawing.Point(44, 19)
        Me.tbBC.Name = "tbBC"
        Me.tbBC.ReadOnly = True
        Me.tbBC.Size = New System.Drawing.Size(59, 20)
        Me.tbBC.TabIndex = 7
        Me.tbBC.TabStop = False
        '
        'Label15
        '
        Me.Label15.AutoSize = True
        Me.Label15.Location = New System.Drawing.Point(8, 22)
        Me.Label15.Name = "Label15"
        Me.Label15.Size = New System.Drawing.Size(24, 13)
        Me.Label15.TabIndex = 6
        Me.Label15.Text = "BC:"
        '
        'tbDE
        '
        Me.tbDE.Location = New System.Drawing.Point(151, 19)
        Me.tbDE.Name = "tbDE"
        Me.tbDE.ReadOnly = True
        Me.tbDE.Size = New System.Drawing.Size(59, 20)
        Me.tbDE.TabIndex = 9
        Me.tbDE.TabStop = False
        '
        'Label16
        '
        Me.Label16.AutoSize = True
        Me.Label16.Location = New System.Drawing.Point(115, 22)
        Me.Label16.Name = "Label16"
        Me.Label16.Size = New System.Drawing.Size(25, 13)
        Me.Label16.TabIndex = 8
        Me.Label16.Text = "DE:"
```

```
        '
        'tbHL
        '
        Me.tbHL.Location = New System.Drawing.Point(263, 19)
        Me.tbHL.Name = "tbHL"
        Me.tbHL.ReadOnly = True
        Me.tbHL.Size = New System.Drawing.Size(59, 20)
        Me.tbHL.TabIndex = 11
        Me.tbHL.TabStop = False
        '
        'Label17
        '
        Me.Label17.AutoSize = True
        Me.Label17.Location = New System.Drawing.Point(227, 22)
        Me.Label17.Name = "Label17"
        Me.Label17.Size = New System.Drawing.Size(24, 13)
        Me.Label17.TabIndex = 10
        Me.Label17.Text = "HL:"
        '
        'tbSP
        '
        Me.tbSP.Location = New System.Drawing.Point(368, 19)
        Me.tbSP.Name = "tbSP"
        Me.tbSP.ReadOnly = True
        Me.tbSP.Size = New System.Drawing.Size(59, 20)
        Me.tbSP.TabIndex = 13
        Me.tbSP.TabStop = False
        '
        'Label18
        '
        Me.Label18.AutoSize = True
        Me.Label18.Location = New System.Drawing.Point(332, 22)
        Me.Label18.Name = "Label18"
        Me.Label18.Size = New System.Drawing.Size(24, 13)
        Me.Label18.TabIndex = 12
        Me.Label18.Text = "SP:"
        '
        'GroupBox4
        '
        Me.GroupBox4.Controls.Add(Me.tbDiasm)
        Me.GroupBox4.Controls.Add(Me.tbIR)
        Me.GroupBox4.Controls.Add(Me.Label20)
        Me.GroupBox4.Controls.Add(Me.tbPC)
        Me.GroupBox4.Controls.Add(Me.Label19)
        Me.GroupBox4.Location = New System.Drawing.Point(19, 278)
        Me.GroupBox4.Name = "GroupBox4"
        Me.GroupBox4.Size = New System.Drawing.Size(441, 54)
        Me.GroupBox4.TabIndex = 3
        Me.GroupBox4.TabStop = False
        '
        'tbPC
        '
        Me.tbPC.Location = New System.Drawing.Point(43, 19)
        Me.tbPC.Name = "tbPC"
        Me.tbPC.ReadOnly = True
        Me.tbPC.Size = New System.Drawing.Size(59, 20)
        Me.tbPC.TabIndex = 9
        Me.tbPC.TabStop = False
        '
        'Label19
        '
        Me.Label19.AutoSize = True
        Me.Label19.Location = New System.Drawing.Point(7, 22)
        Me.Label19.Name = "Label19"
        Me.Label19.Size = New System.Drawing.Size(24, 13)
        Me.Label19.TabIndex = 8
        Me.Label19.Text = "PC:"
        '
        'tbIR
        '
        Me.tbIR.Location = New System.Drawing.Point(150, 19)
```

```
        Me.tbIR.Name = "tbIR"
        Me.tbIR.ReadOnly = True
        Me.tbIR.Size = New System.Drawing.Size(59, 20)
        Me.tbIR.TabIndex = 11
        Me.tbIR.TabStop = False
        '
        'Label20
        '
        Me.Label20.AutoSize = True
        Me.Label20.Location = New System.Drawing.Point(114, 22)
        Me.Label20.Name = "Label20"
        Me.Label20.Size = New System.Drawing.Size(21, 13)
        Me.Label20.TabIndex = 10
        Me.Label20.Text = "IR:"
        '
        'tbDiasm
        '
        Me.tbDiasm.Location = New System.Drawing.Point(229, 19)
        Me.tbDiasm.Name = "tbDiasm"
        Me.tbDiasm.ReadOnly = True
        Me.tbDiasm.Size = New System.Drawing.Size(197, 20)
        Me.tbDiasm.TabIndex = 13
        Me.tbDiasm.TabStop = False
        '
        'GroupBox5
        '
        Me.GroupBox5.Controls.Add(Me.tckBarClock)
        Me.GroupBox5.Controls.Add(Me.lbOriginal)
        Me.GroupBox5.Controls.Add(Me.tbClock)
        Me.GroupBox5.Location = New System.Drawing.Point(20, 338)
        Me.GroupBox5.Name = "GroupBox5"
        Me.GroupBox5.Size = New System.Drawing.Size(442, 64)
        Me.GroupBox5.TabIndex = 4
        Me.GroupBox5.TabStop = False
        Me.GroupBox5.Text = "Clock:"
        '
        'tbClock
        '
        Me.tbClock.AutoSize = True
        Me.tbClock.Location = New System.Drawing.Point(14, 23)
        Me.tbClock.Name = "tbClock"
        Me.tbClock.Size = New System.Drawing.Size(34, 13)
        Me.tbClock.TabIndex = 0
        Me.tbClock.Text = "Clock"
        '
        'lbOriginal
        '
        Me.lbOriginal.AutoSize = True
        Me.lbOriginal.Location = New System.Drawing.Point(111, 23)
        Me.lbOriginal.Name = "lbOriginal"
        Me.lbOriginal.Size = New System.Drawing.Size(34, 13)
        Me.lbOriginal.TabIndex = 1
        Me.lbOriginal.Text = "Clock"
        '
        'tckBarClock
        '
        Me.tckBarClock.Location = New System.Drawing.Point(228, 16)
        Me.tckBarClock.Name = "tckBarClock"
        Me.tckBarClock.Size = New System.Drawing.Size(195, 42)
        Me.tckBarClock.TabIndex = 2
        '
        'FormRegisterState
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(472, 414)
        Me.Controls.Add(Me.GroupBox5)
        Me.Controls.Add(Me.GroupBox4)
        Me.Controls.Add(Me.GroupBox3)
        Me.Controls.Add(Me.GroupBox2)
        Me.Controls.Add(Me.GroupBox1)
```

```
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog
        Me.MaximizeBox = False
        Me.Name = "FormRegisterState"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Register State"
        Me.GroupBox1.ResumeLayout(False)
        Me.GroupBox1.PerformLayout()
        Me.GroupBox2.ResumeLayout(False)
        Me.GroupBox2.PerformLayout()
        Me.GroupBox3.ResumeLayout(False)
        Me.GroupBox3.PerformLayout()
        Me.GroupBox4.ResumeLayout(False)
        Me.GroupBox4.PerformLayout()
        Me.GroupBox5.ResumeLayout(False)
        Me.GroupBox5.PerformLayout()
        CType(Me.tckBarClock, System.ComponentModel.ISupportInitialize).EndInit()
        Me.ResumeLayout(False)

    End Sub
    Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
    Friend WithEvents tbFlag As System.Windows.Forms.TextBox
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents tbS As System.Windows.Forms.TextBox
    Friend WithEvents Label6 As System.Windows.Forms.Label
    Friend WithEvents tbZ As System.Windows.Forms.TextBox
    Friend WithEvents Label5 As System.Windows.Forms.Label
    Friend WithEvents tbAC As System.Windows.Forms.TextBox
    Friend WithEvents Label4 As System.Windows.Forms.Label
    Friend WithEvents tbP As System.Windows.Forms.TextBox
    Friend WithEvents Label3 As System.Windows.Forms.Label
    Friend WithEvents tbCY As System.Windows.Forms.TextBox
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents GroupBox2 As System.Windows.Forms.GroupBox
    Friend WithEvents tbM As System.Windows.Forms.TextBox
    Friend WithEvents Label14 As System.Windows.Forms.Label
    Friend WithEvents tbL As System.Windows.Forms.TextBox
    Friend WithEvents Label13 As System.Windows.Forms.Label
    Friend WithEvents tbH As System.Windows.Forms.TextBox
    Friend WithEvents Label12 As System.Windows.Forms.Label
    Friend WithEvents tbE As System.Windows.Forms.TextBox
    Friend WithEvents Label11 As System.Windows.Forms.Label
    Friend WithEvents tbD As System.Windows.Forms.TextBox
    Friend WithEvents Label10 As System.Windows.Forms.Label
    Friend WithEvents tbC As System.Windows.Forms.TextBox
    Friend WithEvents Label9 As System.Windows.Forms.Label
    Friend WithEvents tbB As System.Windows.Forms.TextBox
    Friend WithEvents Label8 As System.Windows.Forms.Label
    Friend WithEvents tbA As System.Windows.Forms.TextBox
    Friend WithEvents Label7 As System.Windows.Forms.Label
    Friend WithEvents GroupBox3 As System.Windows.Forms.GroupBox
    Friend WithEvents tbSP As System.Windows.Forms.TextBox
    Friend WithEvents Label18 As System.Windows.Forms.Label
    Friend WithEvents tbHL As System.Windows.Forms.TextBox
    Friend WithEvents Label17 As System.Windows.Forms.Label
    Friend WithEvents tbDE As System.Windows.Forms.TextBox
    Friend WithEvents Label16 As System.Windows.Forms.Label
    Friend WithEvents tbBC As System.Windows.Forms.TextBox
    Friend WithEvents Label15 As System.Windows.Forms.Label
    Friend WithEvents GroupBox4 As System.Windows.Forms.GroupBox
    Friend WithEvents tbDiasm As System.Windows.Forms.TextBox
    Friend WithEvents tbIR As System.Windows.Forms.TextBox
    Friend WithEvents Label20 As System.Windows.Forms.Label
    Friend WithEvents tbPC As System.Windows.Forms.TextBox
    Friend WithEvents Label19 As System.Windows.Forms.Label
    Friend WithEvents GroupBox5 As System.Windows.Forms.GroupBox
    Friend WithEvents tbClock As System.Windows.Forms.Label
    Friend WithEvents tckBarClock As System.Windows.Forms.TrackBar
    Friend WithEvents lbOriginal As System.Windows.Forms.Label
End Class
```

```
c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\
    FormRegisterState.vb
*******************************************************************************
Public Class FormRegisterState

    Private Sub FormRegisterState_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms
    .FormClosingEventArgs) Handles Me.FormClosing
        RemoveHandler FormMain.GetMachineState.OnEventRegisterUpdate, AddressOf OnEventRegisterUpdate
        RemoveHandler FormMain.GetMachineState.OnEventFlagsUpdate, AddressOf OnEventFlagsUpdate
        RemoveHandler FormMain.GetMachineState.OnEventPCUpdate, AddressOf OnEventPCUpdate
        RemoveHandler FormMain.GetMachineState.OnEventSPUpdate, AddressOf OnEventSPUpdate

    End Sub

    Private Sub FormRegisterState_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load

        '        tbClock.Text = ClockString(FormMain.GetMachineState.DynamicClock)
        '       lbOriginal.Text = ClockString(FormMain.GetMachineState.GetClockHz())
        tckBarClock.Minimum = MachineState.MinClock
        tckBarClock.Maximum = MachineState.MaxClock
        tckBarClock.TickStyle = TickStyle.None
        tckBarClock.Value = FormMain.GetMachineState.GetClockHz()

        OnEventClockUpdate(FormMain.GetMachineState())
        OnEventFlagsUpdate(FormMain.GetMachineState())
        OnEventPCUpdate(FormMain.GetMachineState().GetPC(), FormMain.GetMachineState())
        OnEventRegisterUpdate(FormMain.GetMachineState())
        OnEventSPUpdate(FormMain.GetMachineState())

        AddHandler FormMain.GetMachineState.OnEventRegisterUpdate, AddressOf OnEventRegisterUpdate
        AddHandler FormMain.GetMachineState.OnEventFlagsUpdate, AddressOf OnEventFlagsUpdate
        AddHandler FormMain.GetMachineState.OnEventPCUpdate, AddressOf OnEventPCUpdate
        AddHandler FormMain.GetMachineState.OnEventSPUpdate, AddressOf OnEventSPUpdate

    End Sub

    Private Function ClockString(ByVal freq As Integer) As String
        If freq < 1000 Then
            Return String.Format("{0} Hz", freq)
        ElseIf freq < 1000000 Then
            Return String.Format("{0} KHz", freq / 1000)
        Else
            Return String.Format("{0} MHz", freq / 1000000)
        End If
    End Function
    Public Sub OnEventPCUpdate(ByVal pc As UInt16, ByRef macState As MachineState)
        Dim ir, s As Integer
        ir = macState.GetIR()
        s = 0

        tbPC.Text = String.Format("{0:X4}", pc)
        tbIR.Text = String.Format("{0:X2}", ir)
        Dim str As String = MachineState.Diassemble(ir, macState.GetMemory(pc + 1), macState.
    GetMemory(pc + 2), s)
        tbDiasm.Text = String.Format("{0}", str)
        OnEventClockUpdate(macState)
    End Sub

    Public Sub OnEventRegisterUpdate(ByRef macState As MachineState)
        tbA.Text = String.Format("{0:X2}", macState.GetRegister(MachineState.Regs.A))
        tbB.Text = String.Format("{0:X2}", macState.GetRegister(MachineState.Regs.B))
        tbC.Text = String.Format("{0:X2}", macState.GetRegister(MachineState.Regs.C))
        tbD.Text = String.Format("{0:X2}", macState.GetRegister(MachineState.Regs.D))
        tbE.Text = String.Format("{0:X2}", macState.GetRegister(MachineState.Regs.E))
        tbH.Text = String.Format("{0:X2}", macState.GetRegister(MachineState.Regs.H))
        tbL.Text = String.Format("{0:X2}", macState.GetRegister(MachineState.Regs.L))
        tbM.Text = String.Format("{0:X2}", macState.GetRegister(MachineState.Regs.M))
        tbBC.Text = String.Format("{0:X2}{1:X2}", macState.GetRegister(MachineState.Regs.B), macState
    .GetRegister(MachineState.Regs.C))
```

```
        tbDE.Text = String.Format("{0:X2}{1:X2}", macState.GetRegister(MachineState.Regs.D), macState↙
    .GetRegister(MachineState.Regs.E))
        tbHL.Text = String.Format("{0:X2}{1:X2}", macState.GetRegister(MachineState.Regs.H), macState↙
    .GetRegister(MachineState.Regs.L))
    End Sub
    Public Sub OnEventSPUpdate(ByRef macState As MachineState)
        tbSP.Text = String.Format("{0:X4}", macState.GetSP())
    End Sub

    Public Sub OnEventFlagsUpdate(ByRef macState As MachineState)
        tbFlag.Text = String.Format("{0:X2}", macState.GetFlagRegs())
        tbAC.Text = macState.FlagAC
        tbCY.Text = macState.FlagCY
        tbP.Text = macState.FlagP
        tbS.Text = macState.FlagS
        tbZ.Text = macState.FlagZ
    End Sub



    Public Sub OnEventClockUpdate(ByRef macState As MachineState)
        lbOriginal.Text = ClockString(macState.GetClockHz())
        If Not (tckBarClock.Value = macState.GetClockHz()) And FormMain.GetMachineState().IsRunning()↙
     = True Then
            tckBarClock.Value = macState.GetClockHz
        End If

        tbClock.Text = ClockString(macState.DynamicClock)
    End Sub



    Private Sub tckBarClock_ValueChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles↙
     tckBarClock.ValueChanged
        Dim bState As Boolean = FormMain.GetMachineState().IsRunning
        FormMain.GetMachineState().SetClockHz(tckBarClock.Value)
        lbOriginal.Text = ClockString(FormMain.GetMachineState().GetClockHz())
        If bState Then
            FormMain.GetMachineState().StartExecution()
        End If
    End Sub
End Class


c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormReport.    ↙
    Designer.vb
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormReport
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
```

```
        Dim resources As System.ComponentModel.ComponentResourceManager = New System.ComponentModel.
    ComponentResourceManager(GetType(FormReport))
        Me.rtbOut = New System.Windows.Forms.RichTextBox()
        Me.SuspendLayout()
        '
        'rtbOut
        '
        Me.rtbOut.BackColor = System.Drawing.Color.Navy
        Me.rtbOut.Dock = System.Windows.Forms.DockStyle.Fill
        Me.rtbOut.Font = New System.Drawing.Font("Consolas", 12.0!, System.Drawing.FontStyle.Bold,
    System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.rtbOut.ForeColor = System.Drawing.Color.White
        Me.rtbOut.Location = New System.Drawing.Point(0, 0)
        Me.rtbOut.Name = "rtbOut"
        Me.rtbOut.ReadOnly = True
        Me.rtbOut.Size = New System.Drawing.Size(736, 210)
        Me.rtbOut.TabIndex = 0
        Me.rtbOut.Text = ""
        '
        'FormReport
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(736, 210)
        Me.Controls.Add(Me.rtbOut)
        Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
        Me.Name = "FormReport"
        Me.Text = "Output Console"
        Me.ResumeLayout(False)

    End Sub
    Friend WithEvents rtbOut As System.Windows.Forms.RichTextBox
End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormReport.vb
*******************************************************************************
Public Class FormReport

    Private Sub FormReport_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.
    FormClosingEventArgs) Handles Me.FormClosing
        outform = Nothing
    End Sub

    Private Sub FormReport_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    MyBase.Load
        outform = Me
    End Sub
    Public Shared MainForm As Form = Nothing
    Private Shared outform As FormReport = Nothing

    Private Sub LoadOutput()
        If outform Is Nothing Then
            outform = New FormReport
            outform.MdiParent = MainForm
            outform.Show()
        End If

    End Sub

    Private Delegate Sub dLoadOutput()

    Private Sub Invoke_LoadOutput()
        Try
            FormMain.Invoke(New dLoadOutput(AddressOf LoadOutput), Nothing)
        Catch ex As Exception

        End Try
    End Sub
    Public Shared Sub Write(ByVal str As String)
```

```
        If outform Is Nothing Then
            Dim ff As New FormReport()
            ff.Invoke_LoadOutput()
            While outform Is Nothing
                ' spin and wait
            End While
        End If

        outform.WriteOut(str)
    End Sub
    Public Shared Sub WriteLine(ByVal str As String)
        Write(str + vbCrLf)
    End Sub

    Public Shared Sub Clear()
        Write("")
        outform.ClearOut()
    End Sub

    Private Sub WriteOut(ByVal str As String)
        FormMain.MakeMeFirst(Me)
        rtbOut.Text += str
    End Sub

    Private Sub ClearOut()
        rtbOut.Clear()
    End Sub
End Class


c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormScreenDis. ↙
    Designer.vb
********************************************************************************
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormScreenDis
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.SuspendLayout()
        '
        'FormScreenDis
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(542, 356)
        Me.Name = "FormScreenDis"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "Text and Graphics Display"
        Me.ResumeLayout(False)

    End Sub
End Class
```

```
c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormScreenDis.vb
********************************************************************************
Public Class FormScreenDis
    Private Shared bmp As Bitmap = Nothing
    Private fnt As New Font("Consolas", 8)

    Private Sub FormScreenDis_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
        Try
            Dim g As Graphics
            If bmp Is Nothing Then
                bmp = New Bitmap(&HFF, &HFF)
                g = Graphics.FromImage(bmp)
                g.FillRectangle(Brushes.Black, New Rectangle(0, 0, bmp.Width, bmp.Height))


            End If
            g = Graphics.FromImage(bmp)
            Dim sz As SizeF = g.MeasureString("X", fnt)
            Dim w, h As Integer
            w = bmp.Width / sz.Width
            h = bmp.Height / sz.Height
            Dim str As String
            str = String.Format("Text [{0} x {1}] and Graphics Display [{2} x {3}]", w, h, bmp.Width,
     bmp.Height)
            Me.Text = str

            Me.SetStyle(ControlStyles.AllPaintingInWmPaint Or _
                        ControlStyles.UserPaint Or _
                        ControlStyles.DoubleBuffer, True)

        Catch ex As Exception

        End Try

    End Sub

    Private Shared pt As New GPoint()

    Structure GPoint
        Public x As Integer
        Public y As Integer
        Public clr As Color
        Public command As Integer
    End Structure

    Private Sub ExecuteCommand(ByVal cmd As Integer)
        pt.command = cmd
    End Sub

    Private Sub ShowData(ByVal val As Integer)
        Try

            Dim cmd As Integer = pt.command

            If (cmd And 1) <> 1 Then
                Exit Sub
            End If

            If ((cmd >> 1) And 1) = 1 Then
                Dim g As Graphics = Graphics.FromImage(bmp)
                Dim br As New SolidBrush(Me.GetMyColor(val))
                Dim rect As New Rectangle(0, 0, bmp.Width, bmp.Height)
                g.FillRectangle(br, rect)
                Me.Invalidate()
                FormMain.MakeMeFirst(Me)
                Exit Sub
```

```
            End If

            If ((cmd >> 4) And 1) = 1 Then
                pt.x = val
            End If
            If ((cmd >> 5) And 1) = 1 Then
                pt.y = val
            End If

            If ((cmd >> 6) And 1) = 1 Then
                pt.clr = GetMyColor(val)
            End If

            If ((cmd >> 2) And 1) = 1 Then
                ' text mode : val is ASCII
                Dim g As Graphics = Graphics.FromImage(bmp)
                Dim str As String = Chr(val And &HFF)
                Dim clr As New SolidBrush(pt.clr)
                Dim sz As SizeF = g.MeasureString(str, fnt)

                Dim pointa As Point
                pointa.X = pt.x * sz.Width
                pointa.Y = pt.y * sz.Height
                g.DrawString(str, fnt, clr, pointa)
                Me.Invalidate()
                FormMain.MakeMeFirst(Me)
                Exit Sub
            End If

            If ((cmd >> 3) And 1) = 1 Then
                ' gfx
                ' If (pt.x < 1) Or (pt.x > (bmp.Width - 1)) Or (pt.y < 1) Or (pt.y > (bmp.Height -    ↙
    1)) Then
                'Exit Sub
                ' End If

                bmp.SetPixel(pt.x, pt.y, pt.clr)
                Me.Invalidate()
                FormMain.MakeMeFirst(Me)
                Exit Sub
            End If

            If ((cmd >> 7) And 1) = 1 Then
                Me.Invalidate()
            End If
        Catch ex As Exception
            ' FormReport.WriteLine("Exception in Screen Display: " + vbCrLf + ex.Message)
        End Try

    End Sub

    Private Function GetMyColor(ByVal clr As Integer) As Color
        Dim rr, gg, bb, aa As Integer
        bb = clr And 3
        gg = (clr >> 2) And 3
        rr = (clr >> 4) And 3
        aa = (clr >> 6) And 3

        rr = rr * 255 / 3
        gg = gg * 255 / 3
        bb = bb * 255 / 3
        aa = aa * 255 / 3

        Return Color.FromArgb(aa, rr, gg, bb)
    End Function

    Private Shared kd As FormScreenDis = Nothing
    Public Shared Sub LoadMe(ByVal p As FormMain)
        If kd Is Nothing Then
            kd = New FormScreenDis()
            kd.MdiParent = p
```

```
            kd.Show()
        End If
    End Sub

    Public Shared Sub OutPort(ByVal p As FormMain, ByVal ch As Integer)
        Try
            LoadMe(p)
            kd.ShowData(ch)
        Catch ex As Exception
            FormReport.WriteLine("Exception in FormScreenDis.OutPort(...): " + ex.Message)
        End Try

    End Sub

    Public Shared Sub OutCommandPort(ByVal p As FormMain, ByVal ch As Integer)
        Try
            LoadMe(p)
            kd.ExecuteCommand(ch)
        Catch ex As Exception
            FormReport.WriteLine("Exception in FormScreenDis.OutPort(...): " + ex.Message)
        End Try

    End Sub


    Private Sub FormScreenDis_FormClosing(ByVal sender As System.Object, ByVal e As System.Windows.  ↙
    Forms.FormClosingEventArgs) Handles MyBase.FormClosing
        kd = Nothing
    End Sub

    Private Sub FormScreenDis_Paint(ByVal sender As System.Object, ByVal e As System.Windows.Forms.  ↙
    PaintEventArgs) Handles MyBase.Paint
        e.Graphics.DrawImage(bmp, New Rectangle(0, 0, Me.Width, Me.Height))
    End Sub
End Class


c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormUsart.    ↙
    Designer.vb
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class FormUsart
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.Label1 = New System.Windows.Forms.Label()
        Me.cbComList = New System.Windows.Forms.ComboBox()
        Me.btnSelect = New System.Windows.Forms.Button()
        Me.Label2 = New System.Windows.Forms.Label()
        Me.lbSelectedCom = New System.Windows.Forms.Label()
        Me.SuspendLayout()
        '
```

```
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Location = New System.Drawing.Point(25, 23)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(56, 13)
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "COM Port:"
        '
        'cbComList
        '
        Me.cbComList.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList
        Me.cbComList.FormattingEnabled = True
        Me.cbComList.Location = New System.Drawing.Point(87, 20)
        Me.cbComList.Name = "cbComList"
        Me.cbComList.Size = New System.Drawing.Size(121, 21)
        Me.cbComList.TabIndex = 1
        '
        'btnSelect
        '
        Me.btnSelect.Location = New System.Drawing.Point(214, 18)
        Me.btnSelect.Name = "btnSelect"
        Me.btnSelect.Size = New System.Drawing.Size(169, 23)
        Me.btnSelect.TabIndex = 2
        Me.btnSelect.Text = "Activate"
        Me.btnSelect.UseVisualStyleBackColor = True
        '
        'Label2
        '
        Me.Label2.AutoSize = True
        Me.Label2.Location = New System.Drawing.Point(32, 71)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(101, 13)
        Me.Label2.TabIndex = 3
        Me.Label2.Text = "Selected COM Port:"
        '
        'lbSelectedCom
        '
        Me.lbSelectedCom.AutoSize = True
        Me.lbSelectedCom.Location = New System.Drawing.Point(165, 72)
        Me.lbSelectedCom.Name = "lbSelectedCom"
        Me.lbSelectedCom.Size = New System.Drawing.Size(33, 13)
        Me.lbSelectedCom.TabIndex = 4
        Me.lbSelectedCom.Text = "None"
        '
        'FormUsart
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(407, 157)
        Me.Controls.Add(Me.lbSelectedCom)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.btnSelect)
        Me.Controls.Add(Me.cbComList)
        Me.Controls.Add(Me.Label1)
        Me.Name = "FormUsart"
        Me.Text = "COM Port: [40h: Data, 41h: Command/Status] "
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents cbComList As System.Windows.Forms.ComboBox
    Friend WithEvents btnSelect As System.Windows.Forms.Button
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents lbSelectedCom As System.Windows.Forms.Label
End Class
```

c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\FormUsart.vb

```
********************************************************************************
Imports System.IO.Ports
Imports System.Threading
Public Class FormUsart
    Private Shared com As SerialPort = Nothing
    Private Shared comMode As Boolean = False

    Private ev As New ManualResetEvent(False)

    Private Sub FormUsart_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    MyBase.Load
        ev.Reset()

        Dim str() As String = SerialPort.GetPortNames()
        For Each strCom As String In str
            Dim strComArr() As Char = strCom.ToCharArray
            Dim strComName As String = "COM"
            For Each strComA As Char In strComArr
                If IsNumeric(strComA) = True Then
                    strComName += strComA
                End If
            Next
            cbComList.Items.Add(strComName)
        Next

        If com Is Nothing Then
            cbComList.SelectedIndex = 0
            lbSelectedCom.Text = ""
        Else
            cbComList.SelectedIndex = cbComList.Items.IndexOf(com.PortName.ToUpper())
            lbSelectedCom.Text = com.PortName.ToUpper()
        End If
    End Sub



    Private Shared kd As FormUsart = Nothing
    Public Shared Sub LoadMe(ByVal p As FormMain)
        If kd Is Nothing Then
            kd = New FormUsart()
            kd.MdiParent = p
            kd.Show()
        End If
    End Sub
    Public Shared Sub OutPortCommand(ByVal p As FormMain, ByVal ch As Integer)
        Try
            If com.IsOpen() = True Then
                com.Close()
            End If

            Dim baud As Integer = ch And &H3
            Dim chLen As Integer = (ch >> 3) And &H3
            Dim enPar As Integer = (ch >> 4) And &H1
            Dim evPar As Integer = (ch >> 5) And &H1
            Dim stopLen As Integer = (ch >> 6) And &H3

            Dim synMode As Integer = stopLen And &H1
            Dim scs As Integer = (stopLen >> 1) And &H1

            If comMode = False Then
                ' mode
                If baud = 0 Then
                    ' sync mode ??????
                Else
                    ' async mode
                    If baud = 1 Then
                        com.BaudRate = 1536 * 1024 / 10
                    ElseIf baud = 2 Then
                        com.BaudRate = 1536 * 1024 / (10 * 16)
                    ElseIf baud = 2 Then
                        com.BaudRate = 1536 * 1024 / (10 * 64)
```

```
                End If

                If stopLen = 0 Then
                    com.StopBits = StopBits.None
                ElseIf stopLen = 1 Then
                    com.StopBits = StopBits.One
                ElseIf stopLen = 2 Then
                    com.StopBits = StopBits.OnePointFive
                ElseIf stopLen = 3 Then
                    com.StopBits = StopBits.Two
                End If

            End If

            If chLen = 0 Then
                com.DataBits = 5
            ElseIf chLen = 1 Then
                com.DataBits = 6
            ElseIf chLen = 2 Then
                com.DataBits = 7
            ElseIf chLen = 3 Then
                com.DataBits = 8
            End If

            If enPar = 1 Then
                If evPar = 0 Then
                    com.Parity = Parity.Odd
                Else
                    com.Parity = Parity.Even
                End If
            End If

        Else
        ' command
            Dim txen, dtr, rxe, sbrk, er, rts, ir, eh As Integer
            txen = ch And &H1
            dtr = (ch >> 1) And &H1
            rxe = (ch >> 2) And &H1
            sbrk = (ch >> 3) And &H1
            er = (ch >> 4) And &H1
            rts = (ch >> 5) And &H1
            ir = (ch >> 6) And &H1
            eh = (ch >> 7) And &H1

            If sbrk = 1 Then
                com.BreakState = True
            Else
                com.BreakState = False
            End If

            If dtr = 1 Then
                com.DtrEnable = True
            Else
                com.DtrEnable = False
            End If

            If rts = 1 Then
                com.RtsEnable = True
            Else
                com.RtsEnable = False
            End If

        End If

        com.Open()
        comMode = True
    Catch ex As Exception

    End Try
End Sub
```

```
    Public Shared Function IntFromBool(ByVal b As Boolean) As Integer
        If b = True Then
            Return 1
        Else
            Return 0
        End If
    End Function

    Public Shared Function InPortCommand(ByVal p As FormMain) As Integer
        Try
            Dim txrdy, rxrdy, txempty, pe, oe, fe, bd, dsr As Integer
            If com.BytesToRead > 0 Then
                txrdy = 1
            Else
                txrdy = 0
            End If
            txrdy = txrdy And IntFromBool(Not (com.CtsHolding))

            If com.BytesToWrite > 0 Then
                txempty = 0
            Else
                txempty = 1
            End If

            If com.BytesToRead > 0 Then
                rxrdy = 1
            Else
                rxrdy = 0
            End If

            bd = IntFromBool(com.BreakState)
            pe = IntFromBool(False)
            oe = IntFromBool(False)
            fe = IntFromBool(False)
            dsr = IntFromBool(com.DsrHolding)
            Dim d As Integer
            d = txrdy
            d = d Or (rxrdy << 1)
            d = d Or (txempty << 2)
            d = d Or (pe << 3)
            d = d Or (oe << 4)
            d = d Or (fe << 5)
            d = d Or (bd << 6)
            d = d Or (dsr << 7)
            d = d And &HFF
            Return d
        Catch ex As Exception

            Return 0
        End Try
    End Function

    Public Shared Sub OutPort(ByVal p As FormMain, ByVal ch As Integer)
        Try
            Dim bt As Byte() = {CByte(ch And &HFF)}
            com.Write(bt, 0, bt.Length)
        Catch ex As Exception
        End Try

    End Sub
    Public Shared Function InPort(ByVal p As FormMain) As Integer
        Try
            Return com.ReadByte()
        Catch ex As Exception

            Return 0
        End Try
    End Function

    Private Sub FormUsart_FormClosing(ByVal sender As System.Object, ByVal e As System.Windows.Forms. ↙
    FormClosingEventArgs) Handles MyBase.FormClosing
```

```
        kd = Nothing
    End Sub

    Public Shared Function PortInWait() As Boolean
        Try
            If Not (com Is Nothing) Then
                Return True
            End If

            If kd Is Nothing Then
                Return False
            End If

            While kd.ev.WaitOne() <> True
                Thread.Sleep(50)
            End While

            kd.ev.Reset()

        Catch ex As Exception

        End Try
        Return True
    End Function

    Private Sub btnSelect_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ↙
    btnSelect.Click
        Try
            If Not (com Is Nothing) Then
                com.Close()
                com.Dispose()
            End If
            com = New SerialPort(cbComList.Items(cbComList.SelectedIndex).ToString)
            com.ReadTimeout = SerialPort.InfiniteTimeout
            com.WriteTimeout = SerialPort.InfiniteTimeout

            com.Open()
            lbSelectedCom.Text = com.PortName.ToUpper
            comMode = False
            ev.Set()
        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try
    End Sub
End Class


c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\HiResTimer.vb
*******************************************************************************
Imports System
Imports System.Runtime.InteropServices
Imports System.ComponentModel
Imports System.Threading

Public Class HiResTimer
    <DllImport("Kernel32.dll")> _
    Private Shared Function QueryPerformanceCounter(ByRef lpPerformanceCounter As Long) As Boolean

    End Function

    <DllImport("Kernel32.dll")> _
    Private Shared Function QueryPerformanceFrequency(ByRef lpFrequency As Long) As Boolean

    End Function

    Private startTime, stopTime As Long
    Private freq As Long

    Public Sub New()
        startTime = 0
        stopTime = 0
```

```
        freq = 0
        If QueryPerformanceFrequency(freq) = False Then
            MessageBox.Show("Hi Performance Timer is not available.....Intel 8085 Clocking will not ↙
    be available", "Hi Performance Timer Absent", MessageBoxButtons.OK, MessageBoxIcon.Stop)
        End If
    End Sub
    Public Sub StartTimer()
        ' Thread.Sleep(0)
        QueryPerformanceCounter(startTime)
    End Sub

    Public Sub StopTimer()
        QueryPerformanceCounter(stopTime)
    End Sub

    Public ReadOnly Property Duration
        Get
            If freq = 0 Then
                Return 0
            Else
                Return CType((stopTime - startTime), Double) / CType(freq, Double)
            End If
        End Get
    End Property


End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\MachineState. ↙
    vb
*******************************************************************************
Imports System.Xml.Serialization
Imports System.Threading
Imports System.Runtime.InteropServices

<Serializable()> <XmlRoot("MachineState")> Public Class MachineState
    'Registers

    Enum Regs As Integer
        B = &H0
        C = &H1
        D = &H2
        E = &H3
        H = &H4
        L = &H5
        M = &H6
        A = &H7
    End Enum

    Private Regs_String() As String = {"B", "C", "D", "E", "H", "L", "M", "A"}

    Public Function GetRegsNameByIndex(ByVal idx As Regs) As String
        Return Regs_String(idx)
    End Function

    Public Function GetRegsIndexFromName(ByVal nm As String) As Regs
        For i = 0 To Regs_String.Length
            If Regs_String(i).Trim.ToUpper = nm.Trim.ToUpper Then
                Return i
            End If
        Next
        Return -1
    End Function


    Private regs_state(8) As Byte
    Public Function GetRegister(ByVal idx As Regs) As Integer
        If idx = Regs.M Then
            Dim addr As Integer = GetRegister(Regs.L)
```

```
            addr = addr Or (GetRegister(Regs.H) << 8)
            regs_state(idx) = GetMemory(addr)
        End If
        Return regs_state(idx)
    End Function
    Public Sub SetRegister(ByVal idx As Regs, ByVal val As Integer, ByVal bRaise As Boolean)
        SyncLock (Me)
            val = val And &HFF
            regs_state(idx) = val
            If idx = Regs.M Then
                Dim addr As Integer = GetRegister(Regs.L)
                addr = addr Or (GetRegister(Regs.H) << 8)
                SetMemory(addr, regs_state(idx), True)
            End If
            Try
                If bRaise = True Then
                    RaiseEvent OnEventRegisterUpdate(Me)
                End If
            Catch ex As Exception

            End Try
        End SyncLock
    End Sub

    ' Clock
    Public Const MinClock As Integer = 1
    Public Const MaxClock As Integer = 2000000

    Private clock_speed As Integer = MaxClock     ' in Hz
    Public Function GetClockHz() As Integer
            Return clock_speed
    End Function
    Public Sub SetClockHz(ByVal sp As Integer)
        SyncLock (Me)
            If sp < MinClock Then
                sp = MinClock
            End If
            If sp > MaxClock Then
                sp = MaxClock
            End If
            clock_speed = sp
        End SyncLock
    End Sub

    Public Function GetTimePeriodInNanoSec() As Double
        Dim t As Double = 1 / GetClockHz()

        Return t
    End Function


    ' Memory
    Private mem(64 * 1024) As Byte
    Public Function GetMemory(ByVal loc As Integer) As Integer
            loc = loc And &HFFFF
            Return mem(loc)
    End Function

    Public Sub SetMemory(ByVal loc As Integer, ByVal val As Integer, ByVal bEv As Boolean)
        SyncLock (Me)
            val = val And &HFF
            loc = loc And &HFFFF
            mem(loc) = val

            Try
                If bEv = True Then
                    RaiseEvent OnEventMemoryUpdate(loc, Me)
                    If loc = GetPC() Then
                        RaiseEvent OnEventPCUpdate(GetPC(), Me)
                    End If
                End If
```

```
            Catch ex As Exception

            End Try
        End SyncLock
    End Sub

    ' Flag Registers
    Private flagsRegs As Byte
    Public Function GetFlagRegs() As Integer
        Return flagsRegs
    End Function
    Public Sub SetFlagRegs(ByVal val As Integer, ByVal bEv As Boolean)
        SyncLock (Me)
            val = val And &HFF
            flagsRegs = val
            Try
                If bEv = True Then
                    RaiseEvent OnEventFlagsUpdate(Me)
                End If
            Catch ex As Exception

            End Try
        End SyncLock
    End Sub

    Public Function GetFlagBit(ByVal bitVal As Integer) As Integer
        bitVal = bitVal And &H7
        Dim f, a, d As Integer
        f = GetFlagRegs()
        a = 1 << bitVal
        f = f And a
        d = f >> bitVal
        d = d And &H1
        Return d
    End Function
    Public Sub SetFlagBit(ByVal bitVal As Integer, ByVal b As Integer)
        bitVal = bitVal And &H7
        b = b And &H1
        Dim f, r, d As Integer
        f = GetFlagRegs()
        r = 1 << bitVal
        d = b << bitVal
        d = d And r

        f = f And Not (r)
        f = f Or d
        SetFlagRegs(f, True)
    End Sub
    ' 1 bit flag set
    <XmlIgnore()> Public Property FlagS As Byte
        Get
            Return GetFlagBit(7)
        End Get
        Set(ByVal value As Byte)
            SetFlagBit(7, value)
        End Set
    End Property

    <XmlIgnore()> Public Property FlagZ As Byte
        Get
            Return GetFlagBit(6)
        End Get
        Set(ByVal value As Byte)
            SetFlagBit(6, value)
        End Set
    End Property

    <XmlIgnore()> Public Property FlagAC As Byte
        Get
            Return GetFlagBit(4)
        End Get
```

```
        Set(ByVal value As Byte)
            SetFlagBit(4, value)
        End Set
    End Property

    <XmlIgnore()> Public Property FlagP As Byte
        Get
            Return GetFlagBit(2)
        End Get
        Set(ByVal value As Byte)
            SetFlagBit(2, value)
        End Set
    End Property

    <XmlIgnore()> Public Property FlagCY As Byte
        Get
            Return GetFlagBit(0)
        End Get
        Set(ByVal value As Byte)
            SetFlagBit(0, value)
        End Set
    End Property

    ' Stack Pointer
    Private sp As UInt16 = &HFFFF
    Public Function GetSP() As Integer
        Return sp
    End Function

    Public Sub SetSP(ByVal val As Integer, ByVal bEv As Boolean)
        val = val And &HFFFF
        sp = val
        Try
            If bEv = True Then
                RaiseEvent OnEventSPUpdate(Me)
            End If
        Catch ex As Exception

        End Try
    End Sub

    ' Program Counter
    <XmlElement("ProgramCounter")> Private pc As UInt16 = 0
    Public Function GetPC() As Integer
        Return pc
    End Function

    Public Sub SetPC(ByVal val As Integer, ByVal bEv As Boolean)
        SyncLock (Me)
            Try
                If bEv = True Then
                    RaiseEvent OnEventPCUpdate(GetPC(), Me)
                End If
            Catch ex As Exception

            End Try
            val = val And &HFFFF
            pc = val

        End SyncLock
    End Sub


    Public Function NextPC(ByVal rel As Integer) As Integer
        Dim val As Integer
        val = GetPC()
        val += rel
        If val < UInt16.MinValue Then
            val += UInt16.MaxValue
        End If
        If val > UInt16.MaxValue Then
```

```
            val -= UInt16.MaxValue
        End If
        If val < UInt16.MinValue Then
            val = UInt16.MinValue
        End If
        SetPC(val, True)
        Return GetPC()
    End Function

    Public Function NextPC() As Integer
        Return NextPC(1)
    End Function

    ' Instruction register
    Public Function GetIR() As Byte
        Return GetMemory(GetPC())
    End Function

    ' Port
    Public Sub InPort(ByVal port As Integer, ByVal bEv As Boolean)
        port = port And &HFF
        Dim data As Byte = 0
        Dim prio As Integer = 0
        Try
            If bEv = True Then
                RaiseEvent OnEventInPortUpdate(data, port, Me, prio)
            End If
        Catch ex As Exception

        End Try
        SetRegister(Regs.A, data, bEv)
    End Sub

    Public Sub OutPort(ByVal port As Integer, ByVal bEv As Boolean)
        port = port And &HFF
        Dim data As Byte
        data = GetRegister(Regs.A)
        Try
            If bEv = True Then
                RaiseEvent OnEventOutPortUpdate(data, port, Me)
            End If
        Catch ex As Exception

        End Try
    End Sub

    ' Event Management
    Public Event OnEventRegisterUpdate(ByRef macState As MachineState)
    Public Event OnEventMemoryUpdate(ByVal addr As UInt16, ByRef macState As MachineState)
    Public Event OnEventFlagsUpdate(ByRef macState As MachineState)
    Public Event OnEventOutPortUpdate(ByVal data As Byte, ByVal port As Byte, ByRef macState As      ↙
    MachineState)
    Public Event OnEventInPortUpdate(ByRef data As Byte, ByVal port As Byte, ByRef macState As      ↙
    MachineState, ByRef prio As Integer) ' higher prio will be able to push in data
    Public Event OnEventSPUpdate(ByRef macState As MachineState)
    Public Event OnEventPCUpdate(ByVal pc As UInt16, ByRef macState As MachineState)
    Public Event OnStateUpdate(ByVal bRunning As Boolean, ByRef macState As MachineState)

    Private Sub SendEventStatus(ByVal bRunning As Boolean)
        SyncLock (Me)
            Try
                RaiseEvent OnStateUpdate(bRunning, Me)
            Catch ex As Exception

            End Try
        End SyncLock
    End Sub

    'XML Serialization functions
    <XmlElement("PC")> Public Property Xml_PC As Integer
        Get
```

```
            Return GetPC()
        End Get
        Set(ByVal value As Integer)
            SetPC(value, True)
        End Set
    End Property

    <XmlElement("SP")> Public Property Xml_SP As Integer
        Get
            Return GetSP()
        End Get
        Set(ByVal value As Integer)
            SetSP(value, True)
        End Set
    End Property

    <XmlElement("CLOCK")> Public Property Xml_Clock As UInt64
        Get
            Return GetClockHz()
        End Get
        Set(ByVal value As UInt64)
            SetClockHz(value)
        End Set
    End Property

    <XmlElement("FLAGS")> Public Property Xml_Flags As Integer
        Get
            Return GetFlagRegs()
        End Get
        Set(ByVal value As Integer)
            SetFlagRegs(value, True)
        End Set
    End Property

    <XmlArray("REGS")> Public Property Xml_Regs As ArrayList
        Get
            Return New ArrayList(regs_state)
        End Get
        Set(ByVal value As ArrayList)
            value.CopyTo(regs_state)
        End Set
    End Property

    <XmlArray("MEMORY")> Public Property Xml_Memory As ArrayList
        Get
            Return New ArrayList(mem)
        End Get
        Set(ByVal value As ArrayList)
            value.CopyTo(mem)
        End Set
    End Property

    ' Internal Construction
    Public Sub New()
        SetRegister(Regs.A, 0, False)
        FlagAC = 0
        FlagCY = 0
        FlagP = 0
        FlagS = 0
        FlagZ = 1
    End Sub

    ' Misc
    Public Sub ClearMemory()
        For i As Integer = 0 To &HFFFF
            If Not (GetMemory(i) = 0) Then
                SetMemory(i, 0, True)
            End If
        Next
    End Sub
```

```
    Dim th As Thread = Nothing
    Public Function IsRunning() As Boolean
        If th Is Nothing Then
            Return False
        Else
            Return True
        End If
    End Function
    Public Sub StartExecution()
        If Not (th Is Nothing) Then
            StopExecution()
        End If
        Try
            dynaClock = GetClockHz()
            th = New Thread(AddressOf ExecutingMachine)
            th.Start()
            SendEventStatus(True)
            NextPC(0)
        Catch ex As Exception
            SendEventStatus(False)

        End Try
    End Sub
    Public Sub StopExecution()
        If th Is Nothing Then
            Exit Sub
        End If

        Try
            th.Abort()
        Catch ex As Exception

        End Try
        Try
            th.Join()
        Catch ex As Exception

        End Try
        th = Nothing
        SendEventStatus(False)
        NextPC(0)
    End Sub


    Private hiresTimer As New HiResTimer()
    Private dynaClock As Integer = 0
    Public ReadOnly Property DynamicClock As Integer
        Get
            Return dynaClock
        End Get
    End Property



    Private Sub ProcessInstruction()
        hiresTimer.StartTimer()
        Dim tStates As Integer = 1
        ExecuteCU(tStates)
        Dim tWait As Double = tStates * GetTimePeriodInNanoSec()
        hiresTimer.StopTimer()
        If hiresTimer.Duration > tWait Then
            Exit Sub
        Else
            While hiresTimer.Duration < tWait
                hiresTimer.StopTimer()
            End While
        End If

        Dim freq As Double = 1 / (hiresTimer.Duration / tStates)
        dynaClock = freq
    End Sub
```

```
    <DllImport("kernel32.dll")> _
    Private Shared Function GetCurrentThread() As IntPtr

    End Function

    <DllImport("kernel32.dll")> _
    Private Shared Function SetThreadAffinityMask(ByVal hThread As IntPtr, ByVal affMash As UIntPtr) ↙
    As UIntPtr

    End Function

    Private Sub ExecutingMachine()
        Try
            If Environment.ProcessorCount > 1 Then
                Dim c_th As IntPtr
                c_th = GetCurrentThread()
                If SetThreadAffinityMask(c_th, &H1) = 0 Then
                    Debug.Print(String.Format("ERROR: Thread {0} cannot be concised to 1 processor", ↙
    c_th))
                End If
            End If

            While True
                ProcessInstruction()
            End While
        Catch ex As Exception

        End Try
    End Sub


    ' IRQ
    Private bIntEnable As Integer = 0
    Private iIntMask As Integer = 0

    Public Property IsInterruptEnabled As Integer
        Get
            Return bIntEnable
        End Get
        Set(ByVal value As Integer)
            bIntEnable = value And &H1
        End Set
    End Property

    Public ReadOnly Property InterruptMask As Integer
        Get
            Return iIntMask
        End Get
    End Property


    Private Sub IntrJump(ByVal addr As Integer, ByVal bSize As Integer, ByVal bForce As Boolean) ' if↙
     bForce = true then bypass EI/DI
        If IsRunning() = False Then
            Exit Sub
        End If
        If IsInterruptEnabled = False Then
            If bForce = False Then
                Exit Sub
            End If
        End If

        StopExecution()

        SavePCToStack(bSize)
        SetPC(addr, True)
        StartExecution()
    End Sub

    Private Function GetIntMaskBit(ByVal bt As Integer) As Integer
```

```
        bt = bt And &H7
        Dim btMask, btVal As Integer
        btMask = 1 << bt
        btVal = InterruptMask And btMask
        btVal = btVal >> bt
        btVal = btVal And &H1
        Return btVal
    End Function


    Public Sub Interrupt_TRAP()
        IntrJump(&H24, 0, True)
    End Sub

    Public ReadOnly Property IsRST7_5Enabled As Integer
        Get
            Return (GetIntMaskBit(3) And (Not (GetIntMaskBit(4)) And GetIntMaskBit(2))) And &H1
        End Get
    End Property

    Public ReadOnly Property IsRST6_5Enabled As Integer
        Get
            Return (GetIntMaskBit(3) And GetIntMaskBit(1)) And &H1
        End Get
    End Property

    Public ReadOnly Property IsRST5_5Enabled As Integer
        Get
            Return (GetIntMaskBit(3) And GetIntMaskBit(0)) And &H1
        End Get
    End Property

    Public Sub Interrupt_RST7_5()
        If IsRST7_5Enabled = 1 Then
            IntrJump(&H3C, 0, False)
        End If
    End Sub
    Public Sub Interrupt_RST6_5()
        If IsRST6_5Enabled = 1 Then
            IntrJump(&H34, 0, False)
        End If
    End Sub
    Public Sub Interrupt_RST5_5()
        If IsRST5_5Enabled = 1 Then
            IntrJump(&H2C, 0, False)
        End If
    End Sub

    Public Sub Interrupt_RST0()
        IntrJump(&H0, 1, False)
    End Sub
    Public Sub Interrupt_RST1()
        IntrJump(&H8, 1, False)
    End Sub
    Public Sub Interrupt_RST2()
        IntrJump(&H10, 1, False)
    End Sub
    Public Sub Interrupt_RST3()
        IntrJump(&H18, 1, False)
    End Sub
    Public Sub Interrupt_RST4()
        IntrJump(&H20, 1, False)
    End Sub
    Public Sub Interrupt_RST5()
        IntrJump(&H28, 1, False)
    End Sub
    Public Sub Interrupt_RST6()
        IntrJump(&H30, 1, False)
    End Sub
    Public Sub Interrupt_RST7()
        IntrJump(&H38, 1, False)
```

```
    End Sub

    Public Sub Interrupt_RST(ByVal r_id As Integer)
        r_id = r_id And &H7

        If r_id = 0 Then
            Interrupt_RST0()
        ElseIf r_id = 1 Then
            Interrupt_RST1()
        ElseIf r_id = 2 Then
            Interrupt_RST2()
        ElseIf r_id = 3 Then
            Interrupt_RST3()
        ElseIf r_id = 4 Then
            Interrupt_RST4()
        ElseIf r_id = 5 Then
            Interrupt_RST5()
        ElseIf r_id = 6 Then
            Interrupt_RST6()
        ElseIf r_id = 7 Then
            Interrupt_RST7()
        End If
    End Sub

    Public Event OnEventSID(ByVal bEnable As Boolean, ByRef val As Integer, ByRef prio As Integer,    ↙
    ByVal macState As MachineState)
    Public Event OnEventSOD(ByVal bEnable As Boolean, ByRef val As Integer, ByVal macState As        ↙
    MachineState)


    Public Sub ReadInterruptMaskToA()
        AsmSid()
        iIntMask = iIntMask And &HFF
        SetRegister(Regs.A, iIntMask, True)
    End Sub
    Public Sub ReadInterruptMaskFromA()
        iIntMask = GetRegister(Regs.A)
        AsmSod()
        iIntMask = iIntMask And &HFF
    End Sub

    Private Sub AsmSid()
        Try
            Dim bEnable As Integer
            bEnable = iIntMask >> 6
            bEnable = bEnable And &H1
            Dim bVal, prio As Integer
            bVal = 0
            prio = 0
            RaiseEvent OnEventSID(CType(bEnable, Boolean), bVal, prio, Me)

            bVal = bVal And bEnable

            If bEnable = 1 Then
                Dim a, b, c As Integer
                a = 1 << 7
                b = iIntMask
                c = Not (a) And b
                a = bVal << 7
                c = c Or a
                c = c And &HFF
                iIntMask = c
            End If
        Catch ex As Exception

        End Try
    End Sub

    Private Sub AsmSod()
        Try
            Dim bEnable, bS As Integer
```

```
            bEnable = iIntMask >> 6
            bS = iIntMask >> 7
            bEnable = bEnable And &H1
            bS = bS And bEnable
            RaiseEvent OnEventSOD(CType(bEnable, Boolean), bS, Me)
        Catch ex As Exception

        End Try
    End Sub

    Public Function GetIntMask() As Integer
        Return (iIntMask And &HFF)
    End Function

    Public Function IsSerialEnabled() As Boolean
        Dim bE As Integer
        bE = iIntMask
        bE = bE >> 6
        bE = bE And &H1
        Return CBool(bE)
    End Function
End Class
```

c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\ ↙
MachineState_Machine.vb

```
********************************************************************************
Partial Class MachineState
    Private Sub ExecuteCU(ByRef t As Integer)
        Dim ins As Byte = GetIR()
        Dim jp As Integer = 0
        Dim op, h, l As Byte
        op = (ins >> 6) And &H3
        h = (ins >> 3) And &H7
        l = ins And &H7
        t = 0

        If op = &H0 Then
            ExecH0(h, l, jp, t)
        ElseIf op = &H1 Then
            ExecH1(h, l, jp, t)
        ElseIf op = &H2 Then
            ExecH2(h, l, jp, t)
        ElseIf op = &H3 Then
            ExecH3(h, l, jp, t)
        End If

        If jp = 0 Then
            ThrowTrap(ins, jp, t)
        End If
        NextPC(jp)
    End Sub
    Private Sub ThrowTrap(ByVal ins As Integer, ByRef iSize As Integer, ByRef t As Integer)
        ' Do nothing
        If (t = 0) And (iSize = 0) Then
            MessageBox.Show(String.Format("INVALID INSTRUCTION {0:X2} at {1:X4}", GetMemory(GetPC()), ↙
     GetPC()))
            iSize = 1
            t = 4
        End If
    End Sub


    Private Sub ExecH0(ByVal h As Byte, ByVal l As Byte, ByRef iSize As Integer, ByRef t As Integer)
        If (h = &H0) And (l = &H0) Then
            ' NOP
            iSize = 1
            t = 4
        ElseIf (h = &H4) And (l = &H0) Then
            ' RIM
```

```
            iSize = 1
            t = 4
            ReadInterruptMaskToA()
        ElseIf (h = &H6) And (l = &H0) Then
            ' SIM
            iSize = 1
            t = 4
            ReadInterruptMaskFromA()
        ElseIf ((h And &H1) = 0) And (l = &H1) Then
            ' LXI [B, D, H, SP], dble
            iSize = 3
            t = 10
            Dim rr As Byte
            rr = (h >> 1) And &H3
            Dim bu, bl, b16 As Integer
            bl = GetMemory(GetPC() + 1)
            bu = GetMemory(GetPC() + 2)
            b16 = bl Or (bu << 8)

            If rr = &H3 Then
                SetSP(b16 And &HFFFF, True)
            Else
                rr *= 2
                SetRegister(CType(rr + 1, Regs), b16 And &HFF, False)
                SetRegister(CType(rr, Regs), (b16 >> 8) And &HFF, True)
            End If

        ElseIf ((h And &H1) = 1) And (l = &H1) Then
            ' DAD [B, D, H, SP]
            iSize = 1
            t = 10
            Dim hl As Integer
            hl = GetRegister(Regs.L)
            hl = hl Or (GetRegister(Regs.H) << 8)

            Dim rp As Integer = 0
            Dim rr As Integer = (h >> 1) And &H3
            If rr = &H3 Then
                rp = GetSP()
            Else
                rr *= 2
                rp = GetRegister(CType(rr + 1, Regs))
                rp = rp Or (GetRegister(CType(rr, Regs)) << 8)
            End If
            hl += rp
            SetRegister(Regs.L, hl And &HFF, False)
            SetRegister(Regs.H, (hl >> 8) And &HFF, True)
            hl = hl >> 16
            FlagCY = hl And &H1
        ElseIf ((h And &H5) = 0) And (l = &H2) Then
            ' STAX [B, D]
            iSize = 1
            t = 7
            Dim addr8, addr16, rr, v8 As Integer
            rr = (h >> 1) And &H1
            rr *= 2
            addr8 = GetRegister(CType(rr + 1, Regs))
            addr16 = addr8
            addr8 = GetRegister(CType(rr, Regs))
            addr16 = addr16 Or (addr8 << 8)
            v8 = GetRegister(Regs.A)
            SetMemory(addr16, v8, True)
        ElseIf ((h And &H5) = 1) And (l = &H2) Then

            ' LDAX [B, D]
            iSize = 1
            t = 7

            Dim addr8, addr16, rr, v8 As Integer
            rr = (h >> 1) And &H1
            rr *= 2
```

```
            addr8 = GetRegister(CType(rr + 1, Regs))
            addr16 = addr8
            addr8 = GetRegister(CType(rr, Regs))
            addr16 = addr16 Or (addr8 << 8)
            v8 = GetMemory(addr16)
            SetRegister(Regs.A, v8, True)
        ElseIf ((h And &H6) = 4) And (l = &H2) Then

            'SHLD / LHLD addr16
            iSize = 3
            t = 16
            Dim addr16, addr8 As Integer
            addr8 = GetMemory(GetPC() + 1)
            addr16 = addr8
            addr8 = GetMemory(GetPC() + 2)
            addr16 = addr16 Or (addr8 << 8)

            If (h And &H1) = 1 Then
                ' LHLD
                SetRegister(Regs.L, GetMemory(addr16), False)
                SetRegister(Regs.H, GetMemory(addr16 + 1), True)
            Else
                ' SHLD
                SetMemory(addr16, GetRegister(Regs.L), True)
                SetMemory(addr16 + 1, GetRegister(Regs.H), True)
            End If

        ElseIf ((h And &H6) = 6) And (l = &H2) Then

            'LDA / STA addr16
            iSize = 3
            t = 13
            Dim addr8, addr16 As Integer
            addr8 = GetMemory(GetPC() + 1)
            addr16 = addr8
            addr8 = GetMemory(GetPC() + 2)
            addr16 = addr16 Or (addr8 << 8)
            If (h And &H1) = 1 Then
                ' LDA
                SetRegister(Regs.A, GetMemory(addr16), True)
            Else
                ' STA
                SetMemory(addr16, GetRegister(Regs.A), True)
            End If

        ElseIf ((h And &H1) = 0) And (l = &H3) Then
            ' INX [B, D, H, SP]
            iSize = 1
            t = 6
            Dim rr, v16, v8 As Integer
            rr = (h >> 1) And &H3
            v16 = 0
            If rr = &H3 Then
                'SP
                v16 = GetSP()
            Else
                rr *= 2
                v8 = GetRegister(CType(rr + 1, Regs))
                v16 = v8
                v8 = GetRegister(CType(rr, Regs))
                v16 = v16 Or (v8 << 8)
            End If
            v16 += 1
            If rr = &H3 Then
                'SP
                SetSP(v16 And &HFFFF, True)
            Else
                v8 = v16 And &HFF
                SetRegister(CType(rr + 1, Regs), v8, False)
                v8 = (v16 >> 8) And &HFF
                SetRegister(CType(rr, Regs), v8, True)
```

```
            End If
        ElseIf ((h And &H1) = 1) And (l = &H3) Then

            ' DCX [B, D, H, SP]
            iSize = 1
            t = 6
            Dim rr, v16, v8 As Integer
            rr = (h >> 1) And &H3
            If rr = &H3 Then
                'SP
                v16 = GetSP()
            Else
                rr *= 2
                v8 = GetRegister(CType(rr + 1, Regs))
                v16 = v8
                v8 = GetRegister(CType(rr, Regs))
                v16 = v16 Or (v8 << 8)
            End If
            v16 -= 1

            If rr = &H3 Then
                'SP
                SetSP(v16 And &HFFFF, True)
            Else

                v8 = v16 And &HFF
                SetRegister(CType(rr + 1, Regs), v8, False)
                v8 = (v16 >> 8) And &HFF
                SetRegister(CType(rr, Regs), v8, True)
            End If

            ElseIf l = &H4 Then

                ' INR [B, C, D, E, H, L, M, A]
                iSize = 1
                t = 4
                If CType(h, Regs) = Regs.M Then
                    t = 10
                End If
                Dim v8, v4 As Integer
                v8 = GetRegister(CType(h, Regs))
                v4 = v8 And &HF
                v8 += 1
                EffectAllFlagsButCYAC(v8)
                SetRegister(CType(h, Regs), v8, True)

                v4 += 1
                FlagAC = (v4 >> 4) And &H1
            ElseIf l = &H5 Then

                ' DCR [B, C, D, E, H, L, M, A]
                iSize = 1
                t = 4
                If CType(h, Regs) = Regs.M Then
                    t = 10
                End If
                Dim v8, v4 As Integer
                v8 = GetRegister(CType(h, Regs))
                v4 = v8 And &HF
                v8 -= 1
                v4 -= 1
                FlagAC = (v4 >> 4) And &H1
                EffectAllFlagsButCYAC(v8)
                SetRegister(CType(h, Regs), v8, True)
            ElseIf l = &H6 Then

                ' MVI [ B, C, D, E, H, L, M, A], val8
                iSize = 2
                t = 7
                If CType(h, Regs) = Regs.M Then
                    t = 10
```

```
            End If
            SetRegister(CType(h, Regs), GetMemory(GetPC() + 1), True)
        ElseIf (h = &H0) And (l = &H7) Then
            ' RLC
            iSize = 1
            t = 4
            ExecRLC()
        ElseIf (h = &H1) And (l = &H7) Then
            ' RRC
            iSize = 1
            t = 4
            ExecRRC()
        ElseIf (h = &H2) And (l = &H7) Then
            ' RAL
            iSize = 1
            t = 4
            ExecRAL()
        ElseIf (h = &H3) And (l = &H7) Then
            ' RAR
            iSize = 1
            t = 4
            ExecRAR()
        ElseIf (h = &H4) And (l = &H7) Then

            ' DAA
            iSize = 1
            t = 4
            Dim v8, v4 As Integer
            v8 = GetRegister(Regs.A)
            v4 = v8 And &HF
            If (FlagAC = 1) Or (v4 > 9) Then
                v8 += &H6
            End If
            v4 = v8 >> 4
            If (FlagCY = 1) Or (v4 > 9) Then
                v8 += &H60
            End If
            FlagCY = (v8 >> 8) And &H1
            v8 = v8 And &HFF
            EffectAllFlagsButCYAC(v8)
            SetRegister(Regs.A, v8, True)
            ElseIf (h = &H5) And (l = &H7) Then

                ' CMA
                iSize = 1
                t = 4
                Dim v8 As Integer
                v8 = GetRegister(Regs.A)
                v8 = Not (v8) And &HFF
                SetRegister(Regs.A, v8, True)
            ElseIf (h = &H6) And (l = &H7) Then

                'STC
                iSize = 1
                t = 4
                FlagCY = 1
            ElseIf (h = &H7) And (l = &H7) Then

                ' CMC
                iSize = 1
                t = 4
                FlagCY = Not (FlagCY) And &H1
            End If


    End Sub
    Private Sub ExecH1(ByVal h As Byte, ByVal l As Byte, ByRef iSize As Integer, ByRef t As Integer)
        If (h = &H6) And (l = &H6) Then
            th = Nothing
            NextPC(0)
            SendEventStatus(False)
```

```
            Throw New Exception()
        Else
            iSize = 1
            t = 4
            Dim val As Byte
            val = GetRegister(CType(l, Regs))
            SetRegister(CType(h, Regs), val, True)
        End If

    End Sub
    Private Sub ExecH2(ByVal h As Byte, ByVal l As Byte, ByRef iSize As Integer, ByRef t As Integer)
        Dim va, vb, vc, vf As Integer

        If h = &H0 Then
            ' ADD [B, C, D, E, H, L, M, A]
            iSize = 1
            t = 4
            If CType(l, Regs) = Regs.M Then
                t = 7
            End If
            va = GetRegister(Regs.A)
            vb = GetRegister(CType(l, Regs))
            vc = va + vb
            EffectAllFlagsButCYAC(vc And &HFF)
            FlagCY = (vc >> 8) And &H1
            SetRegister(Regs.A, vc And &HFF, True)
            va = va And &HF
            vb = vb And &HF
            vc = va + vb
            FlagAC = (vc >> 4) And &H1
        ElseIf h = &H1 Then
            ' ADC [B, C, D, E, H, L, M, A]
            iSize = 1
            t = 4
            If CType(l, Regs) = Regs.M Then
                t = 7
            End If
                 va = GetRegister(Regs.A)
            vb = GetRegister(CType(l, Regs))
            vf = FlagCY
            vc = va + vb + vf
            EffectAllFlagsButCYAC(vc And &HFF)
            FlagCY = (vc >> 8) And &H1
            SetRegister(Regs.A, vc And &HFF, True)
            va = va And &HF
            vb = vb And &HF
            vc = va + vb + vf
            FlagAC = (vc >> 4) And &H1
        ElseIf h = &H2 Then
            'SUB
            iSize = 1
            t = 4
            If CType(l, Regs) = Regs.M Then
                t = 7
            End If
            va = GetRegister(Regs.A)
            vb = GetRegister(CType(l, Regs))
            vc = va - vb
            EffectAllFlagsButCYAC(vc)
            FlagCY = (vc >> 8) And &H1
            SetRegister(Regs.A, vc, True)
            va = va And &HF
            vb = vb And &HF
            vc = va - vb
            FlagAC = (vc >> 4) And &H1
        ElseIf h = &H3 Then
            'SBB
            iSize = 1
            t = 4
            If CType(l, Regs) = Regs.M Then
                t = 7
```

```
            End If
            va = GetRegister(Regs.A)
            vb = GetRegister(CType(l, Regs))
            vf = FlagCY
            vc = va - vb - vf
            EffectAllFlagsButCYAC(vc And &HFF)
            FlagCY = (vc >> 8) And &H1
            SetRegister(Regs.A, vc And &HFF, True)
            va = va And &HF
            vb = vb And &HF
            vc = va - vb - vf
            FlagAC = (vc >> 4) And &H1

        ElseIf h = &H4 Then
            'ANA
            iSize = 1
            t = 4
            If CType(l, Regs) = Regs.M Then
                t = 7
            End If
            va = GetRegister(Regs.A)
            vb = GetRegister(CType(l, Regs))
            vc = va And vb
            vc = vc And &HFFFF

            EffectAllFlagsButCYAC(vc And &HFF)
            FlagCY = 0
            FlagAC = 1
            SetRegister(Regs.A, vc And &HFF, True)

        ElseIf h = &H5 Then
            'XRA
            iSize = 1
            t = 4
            If CType(l, Regs) = Regs.M Then
                t = 7
            End If
            va = GetRegister(Regs.A)
            vb = GetRegister(CType(l, Regs))
            vc = (va And Not (vb)) Or (Not (va) And vb)
            vc = vc And &HFFFF

            EffectAllFlagsButCYAC(vc And &HFF)
            FlagCY = 0
            FlagAC = 0
            SetRegister(Regs.A, vc And &HFF, True)

        ElseIf h = &H6 Then
            'ORA
            iSize = 1
            t = 4
            If CType(l, Regs) = Regs.M Then
                t = 7
            End If
            va = GetRegister(Regs.A)
            vb = GetRegister(CType(l, Regs))
            vc = va Or vb
            vc = vc And &HFFFF

            EffectAllFlagsButCYAC(vc And &HFF)
            FlagCY = 0
            FlagAC = 0
            SetRegister(Regs.A, vc And &HFF, True)
        ElseIf h = &H7 Then
            ' CMP
            iSize = 1
            t = 4
            If CType(l, Regs) = Regs.M Then
                t = 7
            End If
```

```
            va = GetRegister(Regs.A)
            vb = GetRegister(CType(l, Regs))
            vc = va - vb
            EffectAllFlagsButCYAC(vc And &HFF)
            FlagCY = (vc >> 8) And &H1
            va = va And &HF
            vb = vb And &HF
            vc = va - vb
            FlagAC = (vc >> 4) And &H1

        End If


    End Sub
    Private Function VerifyCond(ByVal id As Integer) As Boolean

        If id = 0 Then
            Return (Not (FlagZ) And &H1)
        ElseIf id = 1 Then
            Return FlagZ
        ElseIf id = 2 Then
            Return (Not (FlagCY) And &H1)
        ElseIf id = 3 Then
            Return FlagCY
        ElseIf id = 4 Then
            Return (Not (FlagP) And &H1)
        ElseIf id = 5 Then
            Return FlagP
        ElseIf id = 6 Then
            Return (Not (FlagS) And &H1)
        ElseIf id = 7 Then
            Return FlagS
        End If
        Return 0
    End Function
    Private Sub ExecH3(ByVal h As Byte, ByVal l As Byte, ByRef iSize As Integer, ByRef t As Integer)
        Dim addr, vl, vh As Integer
        Dim va, vb, vc, vf As Integer

        If l = &H0 Then
            ' {"RNZ", "RZ", "RNC", "RC", "RPO", "RPE", "RP", "RM"}
            iSize = 1
            If (VerifyCond(h) And 1) = 1 Then
                t = 12
                iSize = 0
                LoadPCFromStack()
            Else
                t = 6
            End If
        ElseIf ((h And &H1) = 0) And (l = &H1) Then
            '"POP {0}"
            iSize = 1
            t = 10
            vl = 0
            vh = 0
            LoadIntFromStack(vh, vl)

            h >>= 1
            If h = 0 Then
                SetRegister(Regs.B, vh, False)
                SetRegister(Regs.C, vl, True)
            ElseIf h = 1 Then
                SetRegister(Regs.D, vh, False)
                SetRegister(Regs.E, vl, True)
            ElseIf h = 2 Then
                SetRegister(Regs.H, vh, False)
                SetRegister(Regs.L, vl, True)
            Else
                SetRegister(Regs.A, vh, True)
                SetFlagRegs(vl, True)
            End If
```

```
        ElseIf (h = &H1) And (l = &H1) Then
            ' "RET"
            iSize = 0
            t = 10
            LoadPCFromStack()

        ElseIf (h = &H5) And (l = &H1) Then
            '"PCHL"
            iSize = 0
            t = 6
            vl = GetRegister(Regs.L)
            vh = GetRegister(Regs.H)
            addr = vl Or (vh << 8)
            SetPC(addr, True)

        ElseIf (h = &H7) And (l = &H1) Then
            '   "SPHL"
            iSize = 1
            t = 6
            vl = GetRegister(Regs.L)
            vh = GetRegister(Regs.H)
            addr = vl Or (vh << 8)
            SetSP(addr, True)

        ElseIf l = &H2 Then
            '{"JNZ", "JZ", "JNC", "JC", "JPO", "JPE", "JP", "JM"}
            iSize = 3
            If (VerifyCond(h) And 1) = 1 Then
                t = 10
                vl = GetMemory(GetPC() + 1)
                vh = GetMemory(GetPC() + 2)
                addr = vl Or (vh << 8)
                iSize = 0
                SetPC(addr, True)
            Else
                t = 7
            End If

        ElseIf (h = &H0) And (l = &H3) Then
            ' JMP
            iSize = 3
            t = 10
            vl = GetMemory(GetPC() + 1)
            vh = GetMemory(GetPC() + 2)
            addr = vl Or (vh << 8)
            iSize = 0
            SetPC(addr, True)
        ElseIf (h = &H2) And (l = &H3) Then
            '"OUT "
            iSize = 2
            t = 10
            vl = GetMemory(GetPC() + 1)
            OutPort(vl, True)
        ElseIf (h = &H3) And (l = &H3) Then
            'IN
            iSize = 2
            t = 10
            vl = GetMemory(GetPC() + 1)
            InPort(vl, True)
        ElseIf (h = &H5) And (l = &H3) Then
            'XCHG
            iSize = 1
            t = 4
            va = GetRegister(Regs.L)
            vb = GetRegister(Regs.E)
            Swap(va, vb)
            SetRegister(Regs.L, va, False)
            SetRegister(Regs.E, vb, False)

            va = GetRegister(Regs.H)
```

```
            vb = GetRegister(Regs.D)
            Swap(va, vb)
            SetRegister(Regs.H, va, False)
            SetRegister(Regs.D, vb, True)
        ElseIf (h = &H4) And (l = &H3) Then
            '"XTHL"
            iSize = 1
            t = 16

            va = (GetRegister(Regs.H) << 8) Or (GetRegister(Regs.L))
            vb = GetSP()
            Swap(va, vb)
            SetRegister(Regs.L, va And &HFF, False)
            SetRegister(Regs.H, (va >> 8) And &HFF, True)
            SetPC(vb And &HFFFF, True)

        ElseIf (h = &H6) And (l = &H3) Then
            '"DI"
            iSize = 1
            t = 4
            IsInterruptEnabled = False
        ElseIf (h = &H7) And (l = &H3) Then
            '"EI"
            iSize = 1
            t = 4
            IsInterruptEnabled = True
        ElseIf (l = &H4) Then
            '{"CNZ", "CZ", "CNC", "CC", "CPO", "CPE", "CP", "CM"}
            iSize = 3
            If (VerifyCond(h) And 1) = 1 Then
                t = 18
                iSize = 0
                SavePCToStack(3)
                vl = GetMemory(GetPC() + 1)
                vh = GetMemory(GetPC() + 2)
                SetPC((vh << 8) Or vl, True)

            Else
                t = 9
            End If
        ElseIf ((h And &H1) = 0) And (l = &H5) Then
            ' PUSH B, D, H, PSW
            iSize = 1
            t = 12
            h >>= 1
            If h = 0 Then
                SaveIntToStack(GetRegister(Regs.B), GetRegister(Regs.C))
            ElseIf h = 1 Then
                SaveIntToStack(GetRegister(Regs.D), GetRegister(Regs.E))
            ElseIf h = 2 Then
                SaveIntToStack(GetRegister(Regs.H), GetRegister(Regs.L))
            Else
                SaveIntToStack(GetRegister(Regs.A), GetFlagRegs())
            End If
        ElseIf (h = &H1) And (l = &H5) Then
            ' CALL addr16
            iSize = 0
            t = 18
            SavePCToStack(3)
            vl = GetMemory(GetPC() + 1)
            vh = GetMemory(GetPC() + 2)
            addr = (vh << 8) Or vl
            SetPC(addr, True)

        ElseIf (h = &H0) And (l = &H6) Then
            ' ADI byte
            iSize = 2
            t = 7
            va = GetRegister(Regs.A)
            vb = GetMemory(GetPC() + 1)
            vc = va + vb
```

```
            FlagCY = (vc >> 8) And &H1
            vc = vc And &HFF
            SetRegister(Regs.A, vc, True)
            EffectAllFlagsButCYAC(vc)
            va = va And &HF
            vb = vb And &HF
            vc = va + vb
            FlagAC = (vc >> 4) And &H1

        ElseIf (h = &H1) And (l = &H6) Then
            ' ACI byte
            iSize = 2
            t = 7
            va = GetRegister(Regs.A)
            vb = GetMemory(GetPC() + 1)
            vf = GetFlagRegs()
            vc = va + vb + vf
            FlagCY = (vc >> 8) And &H1
            vc = vc And &HFF
            SetRegister(Regs.A, vc, True)
            EffectAllFlagsButCYAC(vc)
            va = va And &HF
            vb = vb And &HF
            vc = va + vb + vf
            FlagAC = (vc >> 4) And &H1

        ElseIf (h = &H2) And (l = &H6) Then
            'SUI
            iSize = 2
            t = 7
            va = GetRegister(Regs.A)
            vb = GetMemory(GetPC() + 1)
            vc = va - vb
            FlagCY = (vc >> 8) And &H1
            vc = vc And &HFF
            SetRegister(Regs.A, vc, True)
            EffectAllFlagsButCYAC(vc)
            va = va And &HF
            vb = vb And &HF
            vc = va - vb
            FlagAC = (vc >> 4) And &H1

        ElseIf (h = &H3) And (l = &H6) Then
            'SBI
            iSize = 2
            t = 7
            va = GetRegister(Regs.A)
            vb = GetMemory(GetPC() + 1)
            vf = GetFlagRegs()
            vc = va - vb - vf
            FlagCY = (vc >> 8) And &H1
            vc = vc And &HFF
            SetRegister(Regs.A, vc, True)
            EffectAllFlagsButCYAC(vc)
            va = va And &HF
            vb = vb And &HF
            vc = va - vb - vf
            FlagAC = (vc >> 4) And &H1

        ElseIf (h = &H4) And (l = &H6) Then
            ' ANI
            iSize = 2
            t = 7
            va = GetRegister(Regs.A)
            vb = GetMemory(GetPC() + 1)
            vc = va And vb
            vc = vc And &HFF
            SetRegister(Regs.A, vc, True)
            EffectAllFlagsButCYAC(vc)
            FlagCY = 0
            FlagAC = 1
```

```
        ElseIf (h = &H5) And (l = &H6) Then
            'XRI
            iSize = 2
            t = 7
            va = GetRegister(Regs.A)
            vb = GetMemory(GetPC() + 1)
            vc = (va And Not (vb)) Or (Not (va) And vb)
            vc = vc And &HFF
            SetRegister(Regs.A, vc, True)
            EffectAllFlagsButCYAC(vc)
            FlagCY = 0
            FlagAC = 0

        ElseIf (h = &H6) And (l = &H6) Then
            'ORI
            iSize = 2
            t = 7
            va = GetRegister(Regs.A)
            vb = GetMemory(GetPC() + 1)
            vc = va Or vb
            vc = vc And &HFF
            SetRegister(Regs.A, vc, True)
            EffectAllFlagsButCYAC(vc)
            FlagCY = 0
            FlagAC = 0

        ElseIf (h = &H7) And (l = &H6) Then
            'CPI
            iSize = 2
            t = 7
            va = GetRegister(Regs.A)
            vb = GetMemory(GetPC() + 1)
            vc = va - vb
            EffectAllFlagsButCYAC(vc And &HFF)
            FlagCY = (vc >> 8) And &H1
            va = va And &HF
            vb = vb And &HF
            vc = va - vb
            FlagAC = (vc >> 4) And &H1

        ElseIf l = &H7 Then
            'RST
            iSize = 1
            t = 12
            Interrupt_RST(h)
        End If

    End Sub

    Private Sub EffectAllFlagsButCYAC(ByVal v8 As Integer)
        v8 = v8 And &HFF
        If v8 = 0 Then
            FlagZ = 1
        Else
            FlagZ = 0
        End If

        FlagS = (v8 >> 7) And &H1

        Dim bP, b1 As Integer
        bP = v8 And &H1
        v8 >>= 1
        While Not (v8 = 0)
            b1 = v8 And &H1
            v8 >>= 1
            bP = (b1 And Not (bP)) Or (Not (b1) And bP)
        End While
        FlagP = (Not (FlagZ) And bP) And &H1

    End Sub
```

```
    Private Sub Swap(ByRef a As Integer, ByRef b As Integer)
        Dim c As Integer
        c = a
        a = b
        b = c
    End Sub

    Private Sub ExecRAL()
        Dim a, cy As Integer
        a = GetRegister(Regs.A)
        cy = FlagCY

        a = a << 1
        a = a Or cy
        cy = (a >> 8) And &H1
        a = a And &HFF

        SetRegister(Regs.A, a, True)
        FlagCY = cy
    End Sub

    Private Sub ExecRAR()
        Dim a, cy As Integer
        a = GetRegister(Regs.A)
        cy = FlagCY

        a = a Or (cy << 8)
        cy = a And &H1
        a = a >> 1
        a = a And &HFF

        SetRegister(Regs.A, a, True)
        FlagCY = cy
    End Sub

    Private Sub ExecRLC()
        Dim a, cy As Integer
        a = GetRegister(Regs.A)

        a = a << 1
        cy = (a >> 8) And &H1
        a = a Or cy
        a = a And &HFF

        SetRegister(Regs.A, a, True)
        FlagCY = cy

    End Sub

    Private Sub ExecRRC()
        Dim a, cy As Integer
        a = GetRegister(Regs.A)

        cy = a And &H1
        a = a Or (cy << 8)
        a = a >> 1
        a = a And &HFF

        SetRegister(Regs.A, a, True)
        FlagCY = cy

    End Sub


    Private Sub Save8bitToStack(ByVal d8 As Integer)
        Dim addr As Integer = GetSP()
        addr -= 1
        SetMemory(addr, d8, True)
        SetSP(addr, True)
    End Sub
```

```
    Private Function Load8bitFromStack()
        Dim addr As Integer = GetSP()
        Dim d8 As Integer
        d8 = GetMemory(addr)
        addr += 1
        SetSP(addr, True)
        Return d8
    End Function

    Private Sub SavePCToStack(ByVal ref As Integer)
        Dim addr As Integer = GetPC()
        addr += ref
        SaveIntToStack(addr >> 8, addr)
    End Sub

    Private Sub LoadPCFromStack()
        Dim addr, vl, vh As Integer
        LoadIntFromStack(vh, vl)
        addr = vl Or (vh << 8)
        SetPC(addr, True)
    End Sub

    Private Sub SaveIntToStack(ByVal vh As Integer, ByVal vl As Integer)
        Save8bitToStack(vh)
        Save8bitToStack(vl)
    End Sub

    Private Sub LoadIntFromStack(ByRef vh As Integer, ByRef vl As Integer)
        vl = Load8bitFromStack()
        vh = Load8bitFromStack()
    End Sub
End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\ ↙
    MachineStateClass_Diassembly.vb
********************************************************************************
Partial Class MachineState

    Private Shared s_reg() As String = {"B", "C", "D", "E", "H", "L", "M", "A"}
    Private Shared s_regp() As String = {"B", "D", "H", "SP"}
    Private Shared s_regp1() As String = {"B", "D", "H", "PSW"}

    Private Shared Function DiassembleH3(ByVal h As Byte, ByVal l As Byte, ByVal op1 As Byte, ByVal ↙
    op2 As Byte, ByRef iSize As Integer) As String
        Dim str As String
        If l = &H0 Then
            Dim s_t() As String = {"RNZ", "RZ", "RNC", "RC", "RPO", "RPE", "RP", "RM"}
            iSize = 1
            str = String.Format("{0}", s_t(h))
            Return str
        ElseIf (h And &H1) = 0 And l = &H1 Then
            iSize = 1
            str = String.Format("POP {0}", s_regp1((h >> 1) And &H3))
            Return str
        ElseIf h = &H1 And l = &H1 Then
            iSize = 1
            Return "RET"
        ElseIf h = &H5 And l = &H1 Then
            iSize = 1
            Return "PCHL"
        ElseIf h = &H7 And l = &H1 Then
            iSize = 1
            Return "SPHL"
        ElseIf l = &H2 Then
            Dim s_t() As String = {"JNZ", "JZ", "JNC", "JC", "JPO", "JPE", "JP", "JM"}
            iSize = 3
            str = String.Format("{0} {1:X2}{2:X2}", s_t(h), op2, op1)
            Return str
```

```
        ElseIf h = &H0 And l = &H3 Then
            iSize = 3
            str = String.Format("JMP {1:X2}{2:X2}", op2, op1)
            Return str
        ElseIf h = &H2 And l = &H3 Then
            iSize = 2
            str = String.Format("OUT {0:X2}", op1)
            Return str
        ElseIf h = &H3 And l = &H3 Then
            iSize = 2
            str = String.Format("IN {0:X2}", op1)
            Return str
        ElseIf h = &H4 And l = &H3 Then
            iSize = 1
            Return "XCHG"
        ElseIf h = &H5 And l = &H3 Then
            iSize = 1
            Return "XTHL"
        ElseIf h = &H6 And l = &H3 Then
            iSize = 1
            Return "DI"
        ElseIf h = &H7 And l = &H3 Then
            iSize = 1
            Return "EI"
        ElseIf l = &H4 Then
            Dim s_t() As String = {"CNZ", "CZ", "CNC", "CC", "CPO", "CPE", "CP", "CM"}
            iSize = 3
            str = String.Format("{0} {1:X2}{2:X2}", s_t(h), op2, op1)
            Return str
        ElseIf (h And &H1) = 0 And l = &H5 Then
            iSize = 1
            str = String.Format("PUSH {0}", s_regp1((h >> 1) And &H3))
            Return str
        ElseIf h = &H1 And l = &H5 Then
            iSize = 3
            str = String.Format("CALL {0:X2}{1:X2}", op2, op1)
            Return str
        ElseIf h = &H0 And l = &H6 Then
            iSize = 2
            str = String.Format("ADI {0:X2}", op1)
            Return str
        ElseIf h = &H1 And l = &H6 Then
            iSize = 2
            str = String.Format("ACI {0:X2}", op1)
            Return str
        ElseIf h = &H2 And l = &H6 Then
            iSize = 2
            str = String.Format("SUI {0:X2}", op1)
            Return str
        ElseIf h = &H3 And l = &H6 Then
            iSize = 2
            str = String.Format("SBI {0:X2}", op1)
            Return str
        ElseIf h = &H4 And l = &H6 Then
            iSize = 2
            str = String.Format("ANI {0:X2}", op1)
            Return str
        ElseIf h = &H5 And l = &H6 Then
            iSize = 2
            str = String.Format("XRI {0:X2}", op1)
            Return str
        ElseIf h = &H6 And l = &H6 Then
            iSize = 2
            str = String.Format("ORI {0:X2}", op1)
            Return str
        ElseIf h = &H7 And l = &H6 Then
            iSize = 2
            str = String.Format("CPI {0:X2}", op1)
            Return str
        ElseIf l = &H7 Then
            iSize = 1
```

```
            str = String.Format("RST {0:X1}", h)
        End If
        Return "UNKNOWN"
    End Function

    Private Shared Function DiassembleH2(ByVal h As Byte, ByVal l As Byte, ByVal op1 As Byte, ByVal ↙
    op2 As Byte, ByRef iSize As Integer) As String
        Dim str As String
        If h = &H0 Then
            iSize = 1
            str = String.Format("ADD {0}", s_reg(l))
            Return str
        ElseIf h = &H1 Then
            iSize = 1
            str = String.Format("ADC {0}", s_reg(l))
            Return str
        ElseIf h = &H2 Then
            iSize = 1
            str = String.Format("SUB {0}", s_reg(l))
            Return str
        ElseIf h = &H3 Then
            iSize = 1
            str = String.Format("SBB {0}", s_reg(l))
            Return str
        ElseIf h = &H4 Then
            iSize = 1
            str = String.Format("ANA {0}", s_reg(l))
            Return str
        ElseIf h = &H5 Then
            iSize = 1
            str = String.Format("XRA {0}", s_reg(l))
            Return str
        ElseIf h = &H6 Then
            iSize = 1
            str = String.Format("ORA {0}", s_reg(l))
            Return str
        ElseIf h = &H7 Then
            iSize = 1
            str = String.Format("CMP {0}", s_reg(l))
            Return str
        End If
        Return "UNKNOWN"
    End Function


    Private Shared Function DiassembleH1(ByVal h As Byte, ByVal l As Byte, ByVal op1 As Byte, ByVal ↙
    op2 As Byte, ByRef iSize As Integer) As String
        Dim str As String
        If h = &H6 And l = &H6 Then
            iSize = 1
            Return "HLT"
        Else
            iSize = 1
            str = String.Format("MOV {0}, {1}", s_reg(h), s_reg(l))
            Return str
        End If

    End Function


    Private Shared Function DiassembleH0(ByVal h As Byte, ByVal l As Byte, ByVal op1 As Byte, ByVal ↙
    op2 As Byte, ByRef iSize As Integer) As String
        Dim str As String
        If h = &H0 And l = &H0 Then
            iSize = 1
            Return "NOP"
        ElseIf h = &H4 And l = &H0 Then
            iSize = 1
            Return "RIM"
```

```
        ElseIf h = &H6 And l = &H0 Then
            iSize = 1
            Return "SIM"
        ElseIf (h And &H1) = 0 And l = &H1 Then
            iSize = 3
            str = String.Format("LXI {0}, {1:X2}{2:X2}", s_regp((h >> 1) And &H3), op2, op1)
            Return str
        ElseIf (h And &H1) = 1 And l = &H1 Then
            iSize = 1
            str = String.Format("DAD {0}", s_regp((h >> 1) And &H3))
            Return str
        ElseIf (h And &H5) = 0 And l = &H2 Then
            iSize = 1
            str = String.Format("STAX {0}", s_regp((h >> 1) And &H1))
            Return str
        ElseIf (h And &H5) = 1 And l = &H2 Then
            iSize = 1
            str = String.Format("LDAX {0}", s_regp((h >> 1) And &H1))
            Return str
        ElseIf (h And &H6) = 4 And l = &H2 Then
            iSize = 3
            If (h And &H1) = 1 Then
                str = "LHLD"
            Else
                str = "SHLD"
            End If
            Return String.Format("{0} {1:X2}{2:X2}", str, op2, op1)
        ElseIf (h And &H6) = 6 And l = &H2 Then
            iSize = 3
            If (h And &H1) = 1 Then
                str = "LDA"
            Else
                str = "STA"
            End If
            Return String.Format("{0} {1:X2}{2:X2}", str, op2, op1)
        ElseIf (h And &H1) = 0 And l = &H3 Then
            iSize = 1
            str = String.Format("INX {0}", s_regp((h >> 1) And &H3))
            Return str
        ElseIf (h And &H1) = 1 And l = &H3 Then
            iSize = 1
            str = String.Format("DCX {0}", s_regp((h >> 1) And &H3))
            Return str
        ElseIf l = &H4 Then
            iSize = 1
            str = String.Format("INR {0}", s_reg(h))
            Return str
        ElseIf l = &H5 Then
            iSize = 1
            str = String.Format("DCR {0}", s_reg(h))
            Return str
        ElseIf l = &H6 Then
            iSize = 2
            str = String.Format("MVI {0}, {1:X2}", s_reg(h), op1)
            Return str
        ElseIf h = &H0 And l = &H7 Then
            iSize = 1
            Return "RLC"
        ElseIf h = &H1 And l = &H7 Then
            iSize = 1
            Return "RRC"
        ElseIf h = &H2 And l = &H7 Then
            iSize = 1
            Return "RAL"
        ElseIf h = &H3 And l = &H7 Then
            iSize = 1
            Return "RAR"
        ElseIf h = &H4 And l = &H7 Then
            iSize = 1
            Return "DAA"
        ElseIf h = &H5 And l = &H7 Then
```

```
            iSize = 1
            Return "CMA"
        ElseIf h = &H6 And l = &H7 Then
            iSize = 1
            Return "STC"
        ElseIf h = &H7 And l = &H7 Then
            iSize = 1
            Return "CMC"
        Else
            Return "UNKNOWN"
        End If
    End Function

    Public Shared Function Diassemble(ByVal id As Byte, ByVal op1 As Byte, ByVal op2 As Byte, ByRef ↙
    iSize As Integer) As String
        Dim op, h, l As Integer
        l = id And &H7
        h = (id And &H38) >> 3
        op = (id And &HC0) >> 6


        If op = &H0 Then
            Return DiassembleH0(h, l, op1, op2, iSize)
        ElseIf op = &H1 Then
            Return DiassembleH1(h, l, op1, op2, iSize)
        ElseIf op = &H2 Then
            Return DiassembleH2(h, l, op1, op2, iSize)
        ElseIf op = &H3 Then
            Return DiassembleH3(h, l, op1, op2, iSize)
        End If

        Return "UNKNOWN"
    End Function

End Class



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\My Project\   ↙
    Application.Designer.vb
******************************************************************************
'------------------------------------------------------------------------------
' <auto-generated>
'     This code was generated by a tool.
'     Runtime Version:4.0.30319.235
'
'     Changes to this file may cause incorrect behavior and will be lost if
'     the code is regenerated.
' </auto-generated>
'------------------------------------------------------------------------------

Option Strict On
Option Explicit On


Namespace My

    'NOTE: This file is auto-generated; do not modify it directly.  To make changes,
    ' or if you encounter build errors in this file, go to the Project Designer
    ' (go to Project Properties or double-click the My Project node in
    ' Solution Explorer), and make changes on the Application tab.
    '
    Partial Friend Class MyApplication

        <Global.System.Diagnostics.DebuggerStepThroughAttribute()>  _
        Public Sub New()
            MyBase.New(Global.Microsoft.VisualBasic.ApplicationServices.AuthenticationMode.Windows)
            Me.IsSingleInstance = false
            Me.EnableVisualStyles = true
            Me.SaveMySettingsOnExit = true
            Me.ShutDownStyle = Global.Microsoft.VisualBasic.ApplicationServices.ShutdownMode.        ↙
```

```
    AfterMainFormCloses
        End Sub

        <Global.System.Diagnostics.DebuggerStepThroughAttribute()>  _
        Protected Overrides Sub OnCreateMainForm()
            Me.MainForm = Global.Intel8085Sim.FormMain
        End Sub
    End Class
End Namespace
```

c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\My Project\ ↙
AssemblyInfo.vb

```
'------------------------------------------------------------------------------
Imports System
Imports System.Reflection
Imports System.Runtime.InteropServices

' General Information about an assembly is controlled through the following
' set of attributes. Change these attribute values to modify the information
' associated with an assembly.

' Review the values of the assembly attributes

<Assembly: AssemblyTitle("Intel8085Sim")>
<Assembly: AssemblyDescription("Intel 8085 Simulator")>
<Assembly: AssemblyCompany("Arnav Mukhopadhyay")>
<Assembly: AssemblyProduct("Intel8085Sim")>
<Assembly: AssemblyCopyright("Copyright © Arnav Mukhopadhyay 2011")>
<Assembly: AssemblyTrademark("Intel 8085 Simulator By Arnav Mukhopadhyay")>

<Assembly: ComVisible(False)>

'The following GUID is for the ID of the typelib if this project is exposed to COM
<Assembly: Guid("01b773f5-ddb9-46de-8be7-0cb31c974650")>

' Version information for an assembly consists of the following four values:
'
'      Major Version
'      Minor Version
'      Build Number
'      Revision
'
' You can specify all the values or you can default the Build and Revision Numbers
' by using the '*' as shown below:
' <Assembly: AssemblyVersion("1.0.*")>

<Assembly: AssemblyVersion("1.0.0.0")>
<Assembly: AssemblyFileVersion("1.0.0.0")>
```

c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\My Project\ ↙
Resources.Designer.vb

```
'------------------------------------------------------------------------------
'------------------------------------------------------------------------------
' <auto-generated>
'     This code was generated by a tool.
'     Runtime Version:4.0.30319.235
'
'     Changes to this file may cause incorrect behavior and will be lost if
'     the code is regenerated.
' </auto-generated>
'------------------------------------------------------------------------------

Option Strict On
Option Explicit On


Namespace My.Resources
```

```
    'This class was auto-generated by the StronglyTypedResourceBuilder
    'class via a tool like ResGen or Visual Studio.
    'To add or remove a member, edit your .ResX file then rerun ResGen
    'with the /str option, or rebuild your VS project.
    '''<summary>
    '''  A strongly-typed resource class, for looking up localized strings, etc.
    '''</summary>
    <Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.
    StronglyTypedResourceBuilder", "4.0.0.0"), _
     Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), _
     Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute(), _
     Global.Microsoft.VisualBasic.HideModuleNameAttribute()> _
    Friend Module Resources

        Private resourceMan As Global.System.Resources.ResourceManager

        Private resourceCulture As Global.System.Globalization.CultureInfo

        '''<summary>
        '''  Returns the cached ResourceManager instance used by this class.
        '''</summary>
        <Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.
    EditorBrowsableState.Advanced)> _
        Friend ReadOnly Property ResourceManager() As Global.System.Resources.ResourceManager
            Get
                If Object.ReferenceEquals(resourceMan, Nothing) Then
                    Dim temp As Global.System.Resources.ResourceManager = New Global.System.Resources
    .ResourceManager("Intel8085Sim.Resources", GetType(Resources).Assembly)
                    resourceMan = temp
                End If
                Return resourceMan
            End Get
        End Property

        '''<summary>
        '''  Overrides the current thread's CurrentUICulture property for all
        '''  resource lookups using this strongly typed resource class.
        '''</summary>
        <Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.
    EditorBrowsableState.Advanced)> _
        Friend Property Culture() As Global.System.Globalization.CultureInfo
            Get
                Return resourceCulture
            End Get
            Set(ByVal value As Global.System.Globalization.CultureInfo)
                resourceCulture = value
            End Set
        End Property
    End Module
End Namespace



c:\usr\programs\my projects\mca project\intel 8085 simulator\intel8085sim\Intel8085Sim\My Project\
    Settings.Designer.vb
******************************************************************************
'------------------------------------------------------------------------------
' <auto-generated>
'     This code was generated by a tool.
'     Runtime Version:4.0.30319.235
'
'     Changes to this file may cause incorrect behavior and will be lost if
'     the code is regenerated.
' </auto-generated>
'------------------------------------------------------------------------------

Option Strict On
Option Explicit On
```

```
Namespace My

    <Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute(), _
     Global.System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.SettingsDesigner.SettingsSingleFileGenerator", "10.0.0.0"), _
     Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrowsableState.Advanced)> _
    Partial Friend NotInheritable Class MySettings
        Inherits Global.System.Configuration.ApplicationSettingsBase

        Private Shared defaultInstance As MySettings = CType(Global.System.Configuration.ApplicationSettingsBase.Synchronized(New MySettings), MySettings)

#Region "My.Settings Auto-Save Functionality"
#If _MyType = "WindowsForms" Then
        Private Shared addedHandler As Boolean

        Private Shared addedHandlerLockObject As New Object

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrowsableState.Advanced)> _
        Private Shared Sub AutoSaveSettings(ByVal sender As Global.System.Object, ByVal e As Global.System.EventArgs)
            If My.Application.SaveMySettingsOnExit Then
                My.Settings.Save()
            End If
        End Sub
#End If
#End Region

        Public Shared ReadOnly Property [Default]() As MySettings
            Get

#If _MyType = "WindowsForms" Then
                   If Not addedHandler Then
                        SyncLock addedHandlerLockObject
                            If Not addedHandler Then
                                AddHandler My.Application.Shutdown, AddressOf AutoSaveSettings
                                addedHandler = True
                            End If
                        End SyncLock
                    End If
#End If
                Return defaultInstance
            End Get
        End Property
    End Class
End Namespace

Namespace My

    <Global.Microsoft.VisualBasic.HideModuleNameAttribute(),  _
     Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),  _
     Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute()>  _
    Friend Module MySettingsProperty

        <Global.System.ComponentModel.Design.HelpKeywordAttribute("My.Settings")>  _
        Friend ReadOnly Property Settings() As Global.Intel8085Sim.My.MySettings
            Get
                Return Global.Intel8085Sim.My.MySettings.Default
            End Get
        End Property
    End Module
End Namespace
```