Translating VBA into Assembly Language

Topics

- to use a computer simulator to see how instructions are stored and executed in both machine code and assembly language
- to translate simple programs written in a VBA-like language into assembly language

Preliminaries

Start the machine simulator program, *SimHYMN* found in your lab directory.

In this lab we are going to explore the connection between code written in a higher level language (VBA) and code written in a lower level language (assembly language). The assembly language we are using is for the SimHYMN simulator discussed in your textbook.

This language has a limited number of operations, so to accomplish some relatively simple tasks in this assembly language, you have to write several lines of code. The exercises you are going to do will all be translating programs written in VB into assembly language programs for the simulator, which you will then be able to assemble and run.

To run or write an Assembly language program, start the SimHYMN simulator program, and select Show Editor from the Assembler menu at the top (You'll probably want to make the Editor window bigger.) You can write your code in the editor, or use the File menu of the Editor window to open the file you would like to use. To run a program, first click the Assemble button on the editor. This action loads the program into the simulator where you can then run it (You may want to increase the CPU speed before or while you run the program). You should click the Assemble button on the editor each time you want to run the program; otherwise, values from previous runs of the program will still be in memory and may affect the output.

> For each checkpoint, you are now required to include one or more ACCEPTABLE screenshots of your program code. File `Lab13_ScreenShots.docx` in your lab folder specifies requirements for the screenshots to include. Include your screenshots in `Lab13_ScreenShots.docx`, save this file often and make sure to submit it in your lab folder for grading purposes.

Instructions

1. Translate the following VBA macro into an assembly language program using the variable names in the VBA exactly as the labels in assembly. Save your program as `EchoValues.as`.

```
Sub EchoValues()
  '
  ' Echo Values Macro
  ' This program will get two values, the width and length,
  ' of a rectangle, from the user and display each of them.
  '
  Dim rectangleWidth As Integer
  Dim rectangleLength As Integer

  rectangleWidth = 0
  rectangleLength = 0

  rectangleWidth = InputBox("Enter rectangle width")
  rectangleLength = InputBox("Enter rectangle length")

  MsgBox("The rectangle width is: " & rectangleWidth)
  MsgBox("The rectangle length is: " & rectangleLength)
End Sub
```

Produce the screenshot(s) described in `Lab13_ScreenShots.docx` for the following checkpoint and include the screenshots in the same file under the appropriate header description. Missing or UNACCEPTABLE screenshots will result in failing the corresponding checkpoint.

Checkpoint 1 (30/100)

- ☐ program correctly echos the values the user input to the output

2. Copy and paste the file `EchoValues.as`. Rename the copy `CalculatePerimeter.as`. Open the file in SymHymn.

   Translate the following VBA macro into an assembly language program using the variable names in the VBA exactly as the labels in assembly.

```
Sub CalculatePerimeter()
  '
  ' Calculate Perimeter Macro
  ' This program will calculate and display the perimeter
  ' of a rectangular shape where the width and length are
  ' obtained from the user.
  '
  Dim rectangleWidth As Integer
  Dim rectangleLength As Integer
  Dim rectanglePerimeter As Integer

  rectangleWidth = 0
  rectangleLength = 0
  rectanglePerimeter = 0

  rectangleWidth = InputBox("Enter rectangle width")
  rectangleLength = InputBox("Enter rectangle length")

  rectanglePerimeter = 2 * rectangleWidth + 2 * rectangleLength

  MsgBox("The rectangle perimeter is: " & rectanglePerimeter)
End Sub
```

Produce the screenshot(s) described in `Lab13_ScreenShots.docx` for the following checkpoint and include the screenshots in the same file under the appropriate header description. Missing or UNACCEPTABLE screenshots will result in failing the corresponding checkpoint.

Checkpoint 2 (55/100)

- ☐ program correctly computes the perimeter of a rectangle based on user input width and length and then outputs the result

3. Copy and paste the file `CalculatePerimeter.as`.
   Rename the copy `CalculatePerimeterWithBoundsCheckingPart1.as`. Open the file in SymHymn.

   Translate the following VBA macro into an assembly language program using the variable names in the VBA exactly as the labels in assembly.

```
Sub CalculatePerimeterWithBoundsCheckingPart1()
  '
  ' Calculate Perimeter Macro
  ' This program will calculate and display the perimeter
  ' of a rectangular shape where the width and length are
  ' obtained from the user. The program will check to make
  ' sure that a meaningful value is input for the width.
  '
  Dim rectangleWidth As Integer
  Dim rectangleLength As Integer
  Dim rectanglePerimeter As Integer

  rectangleWidth = 0
  rectangleLength = 0
  rectanglePerimeter = 0

  rectangleWidth = InputBox("Enter rectangle width")
  rectangleLength = InputBox("Enter rectangle length")

  If rectangleWidth > 0 Then
    rectanglePerimeter = 2 * rectangleWidth + 2 * rectangleLength

    MsgBox("The rectangle perimeter is: " & rectanglePerimeter)
  End If
End Sub
```

Produce the screenshot(s) described in `Lab13_ScreenShots.docx` for the following checkpoint and include the screenshots in the same file under the appropriate header description. Missing or UNACCEPTABLE screenshots will result in failing the corresponding checkpoint.

Checkpoint 3 (75/100)

- ☐ program correctly computes the perimeter of a rectangle based on user input width and length and then outputs the result
- ☐ program only computes a perimeter if user input width is greater than 0, otherwise the program outputs nothing

4. Copy and paste the file `CalculatePerimeterWithBoundsCheckingPart1.as`.
   Rename the copy `CalculatePerimeterWithBoundsCheckingPart2.as`. Open the file in SymHymn.

   Translate the following VBA macro into an assembly language program using the variable names in the VBA exactly as the labels in assembly.

```
Sub CalculatePerimeterWithBoundsCheckingPart2()
  '
  ' Calculate Perimeter Macro
  ' This program will calculate and display the perimeter
  ' of a rectangular shape where the width and length are
  ' obtained from the user. The program will check to make
  ' sure that meaningful values are input for the width
  ' and the length.
  '
  Dim rectangleWidth As Integer
  Dim rectangleLength As Integer
  Dim rectanglePerimeter As Integer

  rectangleWidth = 0
  rectangleLength = 0
  rectanglePerimeter = 0

  rectangleWidth = InputBox("Enter rectangle width")
  rectangleLength = InputBox("Enter rectangle length")

  If rectangleWidth > 0 Then
    If rectangleLength > 0 Then
      rectanglePerimeter = 2 * rectangleWidth + 2 * rectangleLength

      MsgBox("The rectangle perimeter is: " & rectanglePerimeter)
    End If
  End If
End Sub
```

Produce the screenshot(s) described in `Lab13_ScreenShots.docx` for the following checkpoint and include the screenshots in the same file under the appropriate header description. Missing or UNACCEPTABLE screenshots will result in failing the corresponding checkpoint.

Checkpoint 4 (90/100)

- ☐ program correctly computes the perimeter of a rectangle based on user input width and length and then outputs the result
- ☐ program only computes a perimeter if user input width and length are each greater than 0, otherwise the program outputs nothing

5. Copy and paste the file `CalculatePerimeterWithBoundsCheckingPart2.as`.
   Rename the copy `CalculatePerimeterWithBoundsCheckingLoop.as`. Open the file in SymHymn.

   Translate the following VBA macro into an assembly language program using the variable names in the VBA exactly as the labels in assembly.

```
Sub CalculatePerimeterWithBoundsCheckingLoop()
  '
  ' Calculate Perimeter Macro
  ' This program will repeatedly calculate and display the
  ' perimeter of a rectangular shape where the width and
  ' length are obtained from the user. The program will
  ' check to make sure that meaningful values are input for
  ' the width and the length.
  '
  Dim rectangleWidth As Integer
  Dim rectangleLength As Integer
  Dim rectanglePerimeter As Integer
  Dim computeAnotherPerimeter As Integer

  rectangleWidth = 0
  rectangleLength = 0
  rectanglePerimeter = 0
  computeAnotherPerimeter = 1

  Do While computeAnotherPerimeter = 1
    rectangleWidth = InputBox("Enter rectangle width")
    rectangleLength = InputBox("Enter rectangle length")

    If rectangleWidth > 0 Then
      If rectangleLength > 0 Then
        rectanglePerimeter = 2 * rectangleWidth + 2 * rectangleLength

        MsgBox("The rectangle perimeter is: " & rectanglePerimeter)
      End If
    End If

    computeAnotherPerimeter = InputBox("Enter 1 to computer another perimeter")
  Loop
End Sub
```

Produce the screenshot(s) described in `Lab13_ScreenShots.docx` for the following checkpoint and include the screenshots in the same file under the appropriate header description. Missing or UNACCEPTABLE screenshots will result in failing the corresponding checkpoint.

Checkpoint 5 (100/100)

- ☐ program correctly computes the perimeter of a rectangle based on user input width and length and then outputs the result
- ☐ program repeatedly computes a perimeter as long user input width and length are each greater than 0

6. Translate the following VBA macro into an assembly language program using the variable names in the VBA exactly as the labels in assembly. Save your program as `CalculateArea.as`.

```
Sub CalculateArea()
  '
  ' Calculate Area Macro
  ' This program will calculate and display the area
  ' of a rectangular shape where the width and length are
  ' obtained from the user. The program will check to make
  ' sure the meaningful values are input.
  '
  Dim rectangleWidth As Integer
  Dim rectangleLength As Integer
  Dim rectangleArea As Integer

  rectangleWidth = 0
  rectangleLength = 0
  rectangleArea = 0

  rectangleWidth = InputBox("Enter rectangle width")
  rectangleLength = InputBox("Enter rectangle length")

  rectangleArea = rectangleWidth * rectangleLength

  MsgBox("The rectangle area is: " & rectangleArea)
End Sub
```

Produce the screenshot(s) described in `Lab13_ScreenShots.docx` for the following checkpoint and include the screenshots in the same file under the appropriate header description. Missing or UNACCEPTABLE screenshots will result in failing the corresponding checkpoint.

Checkpoint 6 (110/100)

- ☐ program correctly computes the area of a rectangle based on user input width and length and then outputs the result