Translating VBA into Assembly Language

Learning objectives

This exercise is designed to help you learn about (and assess whether you have learned about):

- 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

Preparation

Start the machine simulator program, *SimHYMN*, from your *Downloads* page or from the Computer Science Application Page:

http://www.csbsju.edu/computerscience/curriculum/launch/default.htm

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.

Open the Excel spreadsheet in the lab folder and study and run the sample VBA programs, (macros.) Then go on to the exercises where you will translate each of these VBA programs into Assembly Language programs and run each of them in the SimHYMN CPU simulator. 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.

Part 1

Translate the following VBA program into an assembly language program. Save your program as `perimeter.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, rectangleLength As Integer

  rectangleWidth = 0
  rectangleLength = 0

  rectangleWidth = InputBox("Enter rectangle width", "Width")
  rectangleLength = InputBox("Enter rectangle length", "Length")

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

D level – Echo the values the user to the output

```
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, rectangleLength As Integer
  Dim rectanglePerimeter As Integer

  rectangleWidth = 0
  rectangleLength = 0
  rectanglePerimeter = 0

  rectangleWidth = InputBox("Enter rectangle width", "Width")
  rectangleLength = InputBox("Enter rectangle length", "Length")

  rectanglePerimeter = 2 * rectangleWidth + 2 * rectangleLength

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

C level – Compute using values from the user and display output

---

```
Sub CalculatePerimeterWithBoundsChecking()
  '
  ' 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 the meaningful values are input.
  '
  Dim rectangleWidth As Integer, rectangleLength As Integer
  Dim rectanglePerimeter As Integer

  rectangleWidth = 0
  rectangleLength = 0
  rectanglePerimeter = 0

  rectangleWidth = InputBox("Enter rectangle width", "Width")
  rectangleLength = InputBox("Enter rectangle length", "Length")

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

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

```
Sub CalculatePerimeterWithBoundsChecking()
  '
  ' 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 the meaningful values are input.
  '
  Dim rectangleWidth As Integer, rectangleLength As Integer
  Dim rectanglePerimeter As Integer

  rectangleWidth = 0
  rectangleLength = 0
  rectanglePerimeter = 0

  rectangleWidth = InputBox("Enter rectangle width", "Width")
  rectangleLength = InputBox("Enter rectangle length", "Length")

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

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

B level – C level with bounds checking

```
Sub CalculateAreaWithBoundsChecking()
  '
  ' 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, rectangleLength As Integer
  Dim rectangleArea As Integer

  rectangleWidth = 0
  rectangleLength = 0
  rectangleArea = 0

  rectangleWidth = InputBox("Enter rectangle width", "Width")
  rectangleLength = InputBox("Enter rectangle length", "Length")

  If rectangleWidth > 0 And rectangleLength > 0 Then
    rectangleArea = rectangleWidth * rectangleLength
  End If

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

A level – Multiplication with loop
