CS 410 Binary to Assembly Activity Guidelines and Rubric

Overview

In Modules One and Two, you practiced converting code between C++ and assembly. You also learned some of the attributes of assembly and examined how to access and view binary code. In this activity, you will take the next step by performing a task central to Project One, which is due in Module Five. The coding for this assignment will be performed in Codio. You will convert a series of four binary files to assembly and explain the functionality. You will submit the completed Binary to Assembly Activity Template Word Document. The following resources will help you complete the tasks in this assignment:

- Section 6: Working With Binary Files in the *Guide to Software Reverse Engineering*
- Section 9: Using the Bless Hex Editor in the *Codio Guide*

Prompt

For this activity, you will analyze four binary files. For each binary file, you must address the following rubric criteria:

1. List the binary file name.
    - Use the Binary to Assembly Activity Template to complete this step.
    - These files can be found in the Software Reverse Engineering Playground in the Module Three file folder in Codio.
2. Identify the functions in the binary file.
    - Use the Binary to Assembly Activity Template to complete this step.
    - Use the Bless Hex Editor on the Virtual Desktop to identify the functions.
    - List the identified function names for each binary file.
3. Convert the binary file to assembly code.
    - Decompile the binary file found in the Software Reverse Engineering Playground in the Module Three file folder in Codio.
4. Align the blocks of assembly code with their corresponding function in the binary file.
    - Use the Binary to Assembly Activity Template to complete this step.
5. Explain the functionality of the blocks of assembly code.
    - Use the Binary to Assembly Activity Template to complete this step.

What to Submit

Binary to Assembly Activity Template

This should be submitted as a Word document. Use the Binary to Assembly Activity Template to list the binary file names, identify the functions within the binary files, align the blocks of assembly code with their corresponding function, and explain the functionality of the blocks of assembly code.

Binary to Assembly Activity Rubric

Criteria	Exemplary	Proficient	Needs Improvement	Not Evident	Value
Binary File Names	N/A	Lists the binary file name for each file without error (100%)	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include listing the accurate or complete binary file name for each file (85%)	Does not attempt criterion (0%)	8
Identify Binary Functions	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner (100%)	Identifies the functions in each binary file with no significant errors or omissions (85%)	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include identifying the functions in	Does not attempt criterion (0%)	26

			each binary file with no significant errors or omissions (55%)		
Binary to Assembly Conversion	N/A	Converts the binary into assembly code (100%)	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include converting the binary into assembly code (85%)	Does not attempt criterion (0%)	8
Assembly Code Alignment	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner (100%)	Aligns the blocks of assembly code with their corresponding functions in the binary files with no significant errors or omissions (85%)	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include aligning each line of C++ with the corresponding blocks of assembly with fewer errors (55%)	Does not attempt criterion (0%)	22
Explanation of Functionality	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner (100%)	Explains the functionality of the blocks of assembly code with minimal errors and adequate detail (85%)	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include explaining the functionality of the blocks of assembly code with minimal errors and richer detail (55%)	Does not attempt criterion (0%)	27
Articulation of Response	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner (100%)	Clearly conveys meaning with correct grammar, sentence structure, and spelling, demonstrating an understanding of audience and purpose (85%)	Shows progress toward proficiency, but with errors in grammar, sentence structure, and spelling, negatively impacting readability (55%)	Submission has critical errors in grammar, sentence structure, and spelling, preventing understanding of ideas (0%)	9
				Total:	100%