|  |  |  |
| --- | --- | --- |
| Week | Reverse Engineering Malware | Duration |
| 2 | Introduction to Assembly Language | 30 mins |

Marks allocation: 2/100 for CA tutorial submission

**Lesson Objectives**

* Understand basic assembly language

1. Provide examples of three different instruction mnemonics.

**ADD, SUB, MOV**

2. Explain why the term assembler language is not quite correct.

**Assembler means the program that translates your source code. A more correct term is "assembly language".**

3. Explain the difference between big endian and little endian. Also, look up the origins of this term on the Web.

**Little endian places the least significant bit in position 0, on the right side of the number. Big endian does the opposite.**

4. Why might you use a symbolic constant rather than an integer literal in your code?

**An integer literal, such as 35, has no direct meaning to someone reading the program's source code. Instead, a symbolic constant such as MIN\_CAPACITY can be assigned an integer value, and is self-documenting.**

5. How are data labels and code labels different?

**Data labels exist in the data segment as variable offsets. Code labels are in the code segment, and are offsets for transfer of control instructions.**

6. (**True**/False): An identifier cannot begin with a numeric digit.

7. (True/**False**): A hexadecimal literal may be written as 0x3A. **(this notation is used in C, but not in assembly language)**

8. (True/**False**): Assembly language directives execute at runtime.

9. (**True**/False): Assembly language directives can be written in any combination of uppercase and lowercase letters.

10. Name the four basic parts of an assembly language instruction.

**Label, mnemonic, operand(s), comment**

11. (**True**/False): MOV is an example of an instruction mnemonic.

12. (**True**/False): A code label is followed by a colon (:), but a data label does not end with a colon.

END