**Assignment 4 : (Camille Bordes) :**

The received hexadecimal number in an x86 assembly program is a representation of the entered floating-point number because it reflects how the IEEE 754 floating-point standard specifies the encoding and interpretation of floating-point values in memory. Let's break down why the hexadecimal representation corresponds to the entered floating-point number:

1. **IEEE 754 Standard**: The IEEE 754 standard is a widely accepted and standardized way to represent floating-point numbers. It defines the structure of how floating-point values are stored in memory and how they can be interpreted.
2. **Hexadecimal Representation**: In assembly language, memory is typically viewed and manipulated as a series of binary digits (bits). However, when you inspect the memory content, it's more human-readable to display these bits in hexadecimal form.
3. **Bit-Level Storage**: Floating-point numbers, whether single-precision or double-precision, are stored in memory as a sequence of bits. These bits are divided into a sign bit (indicating positive or negative), an exponent (representing the scale of the number), and a significand (holding the fractional part). The IEEE 754 standard specifies the bit-level format.
4. **Hexadecimal Digits**: Each hexadecimal digit corresponds to a group of four bits. Therefore, when you view the memory content in hexadecimal, each digit represents a nibble, which makes it easier to examine the bits.
5. **Interpreting the Hexadecimal Representation**: To interpret the hexadecimal representation as the entered floating-point number, you must consider the following:
   * The first digit represents the sign bit (0 for positive, 1 for negative).
   * The next group of hexadecimal digits represents the exponent, with each hexadecimal digit corresponding to four bits. You need to convert these bits to a decimal value and adjust for bias if applicable.
   * The remaining hexadecimal digits represent the significand, which is typically displayed as a hexadecimal fraction. You can convert this fraction to its binary representation.

By understanding the IEEE 754 standard and the bit-level storage of floating-point values, you can examine the hexadecimal representation in memory and, with the proper interpretation, reconstruct the original floating-point number. This approach allows for consistent and precise representation and interpretation of floating-point values across different systems and programming languages.