Develop an x86-64 program that can perform XOR Fibonacci Encryption/Decryption.

In an XOR encryption system, each character in a string is XOR-ed with a key. The resulting encrypted string can simply be decrypted by performing the XOR operation with the same key.

For example, given the input text “Hello” and the key (in hex) “1A 2B 3C 4D 5E”:

Encryption process

Input Text = **48 65 6C 6C 6F** (“*Hello*” in ASCII)

Key = **1A 2B 3C 4D 5E**

Encrypted Text = **52 4E 50 21 31** (“*RNP!1*” in ASCII)

Decryption process

Input Text = **52 4E 50 21 31** (“*RNP!1*” in ASCII)

Key = **1A 2B 3C 4D 5E**

Decrypted Text = **48 65 6C 6C 6F** (“*Hello*” in ASCII)

Observe how the key is given, and how it must remain consistent between encryption and decryption processes.

In an XOR Fibonacci encryption, the key depends on the Fibonacci sequence. The Fibonacci sequence is a series of numbers where each number is the sum of the two preceding numbers. Typically, the sequence begins with the numbers 0 and 1. However, in this case, let the user decide on the two starting numbers, **n0** and **n1**, both of which are to be declared in memory and limited to the size of a byte. Throughout the sequence, let each element in the sequence be truncated to 8 bits. Lastly, the length of the sequence to be generated should match the length of the input text to be encrypted/decrypted (not including the null terminating character).

Note: Do not include the null terminating character as part of the encryption and decryption process.

For example, let **n0** and **n1** be **74** and **108**, respectively (in decimal), where the input text is “LBYARCH”. Since the input text is seven characters long, the key is “**4A 6C B6 22 D8 FA D2**” (in hex) since:

Sequence Number	Sequence Value (in DEC)	Sequence Value (in HEX)
1	74	4A
2	108	6C

3	182	B6
4	34	22
5	216	D8
6	250	FA
7	210	D2

To summarize the requirements of this program, let the user define the following in memory: input string, n0, and n1. You may assume that the length of the input string is limited to 20 characters. Let the program provide an output following the format below:

```
     Input Text (in ASCII)
     Input Text (in HEX, each character separated by a space)
     Key (in HEX, each character separated by a space)
     Encrypted Text (in Hex, each character separated by a
space)
     Encrypted Text (in ASCII)
```

You may refer to the following examples:

Input (in memory):

Text = “Assembly”
n0 = 25
n1 = 138

Output:

```
Assembly
41 73 73 65 6D 62 6C 79
19 8A A3 2D D0 FD CD CA
58 F9 D0 48 BD 9F A1 B3
XùÐH½Ÿi³
```

Rubric:

Program compiled correctly (but not related to the specs)	0
Program compiled correctly	50
if compiled correctly:	

Printed Input Text in ASCII and in HEX	5
Printed Key in HEX	20
Printed Encrypted text in HEX	20
Printed Encrypted text in ASCII	5
Note: If error in compilation	score = 0
Note: Not following direction (NFD)	minimum deduction starts at 5 points