Hardware e Software das Tecnologias de Informação

Theme 27 – Breaking a Caesar Cypher

Professor: Victor Lobo

Alunos: Sebastião Perestrello (r2014525) e Miguel Duarte (r2011267)

Content

[**Objective** 3](#_Toc439539814)

[**Flowchart** 4](#_Toc439539815)

[**Memory Map** 5](#_Toc439539816)

[**Code with comments** 6](#_Toc439539817)

[**Conclusion** 7](#_Toc439539818)

# **Objective**

The objective of this assignment was to create a program using the instruction set of the 8085 microprocessor that was capable of reading a message that was pre-determined to be encrypted with a Caesar cypher and decode it. Without any kind of clues as to what the message could contain, this would have been a rather complex and somewhat unreliable program to create as frequency analysis may not always be the right way to go around a decryption unless someone is paying attention and confirming the results. As such, it was also provided to the students that the acronym “OTA,” the Air Force base that was to be target of the hypothetical terrorist’s attack, would certainly be included in the message, giving the students something they could more easily use. As the distance between each of the letters remains the same in a Caesar cypher, it is by this acronym that one can determine the encryption key and in this way decode the message, saving the real OTA Air Force base from the hypothetical attack.

# **Flowchart**

As you can see, you have here the flowchart of how the program should function.

It reads the following letter and checks whether it is still within the boundaries of the message. If not, the program executes the exiting instructions; otherwise, it keeps going through the message until it finds a trio of letters that follow the same sequence as the acronym OTA, at which point the decryption key is calculated and the message loop starts anew, but now rewriting the message while applying the decryption key.

Read next letter

Comparison to second part of key sequence

Comparison to first part of key sequence

No

Decrypt letter

Yes

End of message?

Decryption loop

No

No

Set the Caesar shift

Yes

End program

Yes

Is the result 0?

Yes

No

Is the result 0?

End of message?

# **Memory Map**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mem. Address | Label | Mnemonics | Op code | Bytes |
| C000 | START | LDA C04F | 3A | 3 |
| C003 |  | LXI H, C050 | 21 | 3 |
| C006 |  | MVI D, FB | 16 | 2 |
| C008 |  | MVI E, 13 | 1E | 2 |
| C00A |  | MOV C, A | 4F | 1 |
| C00B |  | PUSH B | C5 | 1 |
| C00C |  | PUSH H | E5 | 1 |
| C00D | LOOPSTART | CALL SUBTRACTION | CD | 3 |
| C010 |  | CMP D | BA | 1 |
| C011 |  | JNZ LOOPSTART | C2 | 3 |
| CO14 |  | CALL SUBTRACTION | CD | 3 |
| C017 |  | CMP E | BB | 1 |
| C018 |  | JNZ LOOPSTART | C2 | 3 |
| C01B |  | JMP SETCAESARSHIFT | C3 | 3 |
| C01E | SUBTRACTION | MOV A, M | 7E | 1 |
| C01F |  | INX H | 23 | 1 |
| C020 |  | DCR C | 0D | 1 |
| C021 |  | JM END | FA | 3 |
| C024 |  | SUB M | 96 | 1 |
| C025 |  | RET | C9 | 1 |
| C026 | SETCAESARSHIFT | MOV A, M | 7E | 1 |
| C027 |  | SBI 41 | DE | 2 |
| C029 |  | POP H | E1 | 1 |
| C02A |  | POP B | C1 | 1 |
| C02B |  | MOV B, A | 47 | 1 |
| C02C | DECRYPTIONLOOP | CALL DECRYPT | CD | 3 |
| C02F |  | INX H | 23 | 1 |
| C030 |  | DRC C | 0D | 1 |
| C031 |  | JZ END | CA | 3 |
| C034 |  | JNZ DECRYPTIONLOOP | C2 | 3 |
| C037 | DECRYPT | MOV A, M | 7E | 1 |
| C038 |  | SUB B | 90 | 1 |
| C039 |  | MOV M, A | 77 | 1 |
| C03A |  | RET | C9 | 1 |
| C03B | END | MOV A, B | 78 | 1 |
| C03C |  | HLT | 76 | 1 |

# **Code with comments**

|  |  |  |
| --- | --- | --- |
| Mem. Address | Mnemonic | Comment |
| C000 | LDA C04F | Loads into Acc the length of the message |
| C003 | LXI H, C050 | Loads into HL the address of the 1st letter |
| C006 | MVI D, FB | Move to D the result of (O – T) |
| C008 | MVI E, 13 | Move to E the result of (T – A) |
| C00A | MOV C, A | Saves into C (counter) the size of the message |
| C00B | PUSH B | Saves the initial value in the stack for later use |
| C00C | PUSH H | Saves the initial message address for later use |
| C011 | JNZ LOOPSTART | If the difference between letters isn’t 0 |
| C01B | JMP SETCAESARSHIFT | Conditions met, so set the shift |
| C01E | MOV A, M | Moves into the Acc the current letter |
| C01F | INX H | Increments the address in HL to read the next letter |
| C021 | JM END | If the message is over, jump to the end |
| C024 | SUB M | Acc – letter in HL |
| C027 | SBI 41 | Calculate (letter – A) |
| C029 | POP H | Grab the initial value of the message |
| C02A | POP B | Grab the initial value of the counter |
| C02B | MOV B, A | Move to B the result, making it the current shift |
| C031 | JZ END | If the message is over, jump to the end |
| C034 | JNZ DECRYPTIONLOOP | Else, continue the loop |
| C038 | SUB B | Subtract the shift to Acc so the letter reverts to decrypt it |
| C039 | MOV M, A | Return to M the decrypted letter |
| C03B | MOV A, B | Places into the Acc the shift that was used |

# **Conclusion**

We had picked this project out of the ones available because it sounded, by far, as most interesting. And while it was, in fact, interesting, we can’t ignore the feeling that we have done something incredibly wrong in this assignment. We found it suspiciously easy to complete. We had expected to spend hours trying to wrap our heads around how one would get the message loop to properly detect the acronym the way we wanted, or something similar, especially considering this was one of the projects that would allow someone to get the maximum grade. Strangely enough, it didn’t take one of us more than a few hours of focus to get through it. He even barely had to change the code when he started testing it the following day because all he had forgotten to do was a couple of return statements and to correct a typo.

All we can conclude is that we are either doing something terribly wrong, overlooking a critical detail that would make this the challenge we were expecting (which we find it to be the most likely of the options), or we severely overestimated the complexity of the assignment. Hopefully the latter option is the correct one, but we don’t hold out much hope.