Uniwersytet Przyrodniczo-Humanistyczny w Siedlcach

Marta K. Informatyka II rok

DOKUMENTACJA SAMODZIELNEGO PROJEKTU Z PRZEDMIOTU PROGRAMOWANIE NISKOPOZIOMOWE

Temat: Szyfr One-Time Pad

Spis Treści:

1. Wyjaśnienie szyfru One-Time Pad z wykorzystaniem algorytmu Vigenère'a

Szyfr z kluczem jednorazowym (one-time pad) - szyfr zaproponowany w 1917 roku przez Gilberta Vernama, którego, przy poprawnym wykorzystaniu, nie można złamać. Szyfr z kluczem jednorazowym jest dużym zbiorem o niepowtarzalnych i przypadkowych sekwencjach znaków. Nadawca używa każdej litery z tego zbioru do zaszyfrowania jednego znaku tekstu jawnego.

Algorytm Vigenère'a- jest jednym z klasycznych algorytmów szyfrujących. Należy on do grupy tzw. polialfabetycznych szyfrów podstawieniowych.

Szyfr Vigenère'a może być szyfrem nie do złamania (zostało to udowodnione w 1949 przez Claude'a Elwooda Shannona) przy zachowaniu trzech reguł:

- klucz użyty do szyfrowania wiadomości musi być dłuższy lub równy szyfrowanej wiadomości,
- klucz musi być wygenerowany w sposób całkowicie losowy (nie może istnieć sposób na odtworzenie klucza na podstawie znajomości działania generatorów liczb pseudolosowych),
- klucz nie może być użyty do zaszyfrowania więcej niż jednej wiadomości.

Działanie szyfru oparte jest na takiej tablicy:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Tekst szyfrujemy na podstawie hasła. Szyfrowanie odbywa się w sposób następujący:

Przypuśćmy, że chcemy zaszyfrować prosty tekst, np.:
TO JEST BARDZO TAJNY TEKST

Do tego celu użyjemy znanego tylko nam słowa kluczowego, np. TAJNE
Na początku zauważamy, że użyte słowo kluczowe jest zbyt krótkie, by wystarczyło do zaszyfrowania całego tekstu, więc należy użyć jego wielokrotności. Będzie to miało następującą postać:

TO JEST BARDZO TAJNY TEKST
TA JNET AJNETA JNETA JNETA

Następnie wykonujemy szyfrowanie w następujący sposób: litera szyfrogramu odpowiada literze z tabeli znajdującej się na przecięciu wiersza, wyznaczanego przez literę tekstu jawnego i kolumny wyznaczanej przez literę słowa kluczowego, np. po kolei T i T daje M, O i A daje O itd. W efekcie otrzymujemy zaszyfrowany tekst:

MO SRWM BJEHSO CNNGY CROLT

2. Założenia projektowe

Głównymi założeniami projektu było zaimplementowanie skutecznego i niemożliwego do złamania szyfru typu One-Time Pad, w swoim projekcie wybrałam wersję znakową tego algorytmu, tj Algorytm Vigenere'a, korzystającego z przesunięcia Cezara do przesuwania wiadomości względem klucza. Implementacja spełnia założenia opisane powyżej,tj:

-Klucz jest jednorazowy, może być wygenerowany losowo, klucz może być dłuższy nić wiadomość do zaszyfrowania,

co przy odpowiednim wykorzystaniu algorytmu czyni zaszyfrowany tekst niemożliwym do odczytania bez posiadania klucza szyfrującego.

3.Co umożliwia projekt

-Wpisanie własnego tekstu z klawiatury do tablicy w programie, gdzie zostanie zaszyfrowany zgodnie z algorytmem szyfrowania,

-Wpisanie własnego klucza szyfrującego wiadomość,

-Możliwa implementacja losowania klucza szyfrującego za pomocą generatora liczb pseudolosowych wykorzystujących ziarno (seed),

-Wyniki programu, tj. zaszyfrowana wiadomość i klucz szyfrujący są zwracane do dwóch oddzielnych plików o rozszerzeniu txt, przez co są łatwe do odczytania i przesłania dalej ale trudne do odszyfrowania gdy odbiorca nie posiada dwóch plików, przez co wiadomość jest bezpieczna z dala od niepowołanych osób.

4.Przyjęte ograniczenia

Z uwagi na rodzaj szyfrowania i możliwości pamięci zostały przyjęte następujące ograniczenia:

-znaki wprowadzane z klawiatury są małymi literami alfabetu, bez znaków specjalnych,

-długość ciągu znaków nie przekracza 255

Ograniczenia te nie wpływają jednak na złożoność szyfrowania i jego niezawodność, przez co szyfr nie traci na swej wartości i użyteczności.

5. Rozwiązania Implementacyjne

Rozróżniamy dwa podstawowe rodzaje szyfru One-Time Pad:

-algorytm binarny XOR,

-algorytm znakowy z wykorzystaniem algorytmu Vigenere'a

W swoim projekcie skupiłam się na rozwiązaniu znakowym, ponieważ na poziomie liczb binarnych i kodach ASCII prezentuje się on ciekawiej niż dość prosty algorytm XOR.

Przy implementacji projektu wykorzystywałam biblioteki winapi:

-Kernel32.lib,

-masm32.lib.

Ważniejsze rodzaje zmiennych i rejestrów wykorzystywane w projekcie:

-tablice 16- i 32-bitowe,

-pomocnicze zmienne 16-bitowe,

-uchwyty 32-bitowe,

-rejestry EAX ,EBX, AL, EDX, ECX, ESI i EDI

Wykorzystywane procedury:

```
GetStdHandle PROTO :DWORD
WriteConsoleA PROTO :DWORD,:DWORD,:DWORD,:DWORD,:DWORD
ReadConsoleA  PROTO :DWORD,:DWORD,:DWORD,:DWORD,:DWORD
ExitProcess PROTO :DWORD
wsprintfA PROTO C :VARARG                    ; prototyp procedury w masm32
CreateFileA PROTO :DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD
WriteFile PROTO :DWORD,:DWORD,:DWORD,:DWORD,:DWORD
```

Główny algorytm szyfrujący STEP by STEP:

```
;SZYFROWANIE MOTODA VIGENERE'A

    mov EBX, rin
    sub EBX, 2
    mov pom, EBX
    mov EBX, 0
    mov EDX, 0
    mov EAX, 0
        .WHILE EBX < pom

            mov EAX, 0
            mov EAX, wiadomosc[EBX]
            mov EDX, 0d
            mov EDX, klucz2[EBX]
            sub EDX, 97d
            add EAX, EDX
                .IF AL > 7Ah

                    sub EAX, 7Ah
                    add EAX, 61h
                    mov pom2, EAX ; w pom mam zakodowany kod ascii
                    mov zakodowane[EBX], AL

                .ENDIF
```

1.Do rejestru EBX kopiowana jest ilość znaków wiadomości do zaszyfrowania, odejmowane są ostatnie dwa znaki, a otrzymana realna wartość jest kopiowana do pomocniczej zmiennej pom

2.zerowane są inne potrzebne rejestry,

3.pętla .WHILE wykonuje się tyle razy ile znaków ma wiadomość,

4.z tablicy wiadomość brane po kolei wartości znaków ASCII wiadomości,

5.odejmowana jest wartość 97d, ponieważ od tej liczby zaczynają się pożądane przez nas znaki w tablicy ASCII,

6.do wartości znaku z tablicy wiadomości dodawana jest wartość znaku z tablicy klucz, wynik zapisywany w rejestrze EAX,

7.jeśli wartość w tym rejestrze jest większa niż 122d wchodzimy w instrukcję .IF, musimy tak zrobić, aby znaki przesuwały się zgodnie z algorytmem który jest zaimplementowany,

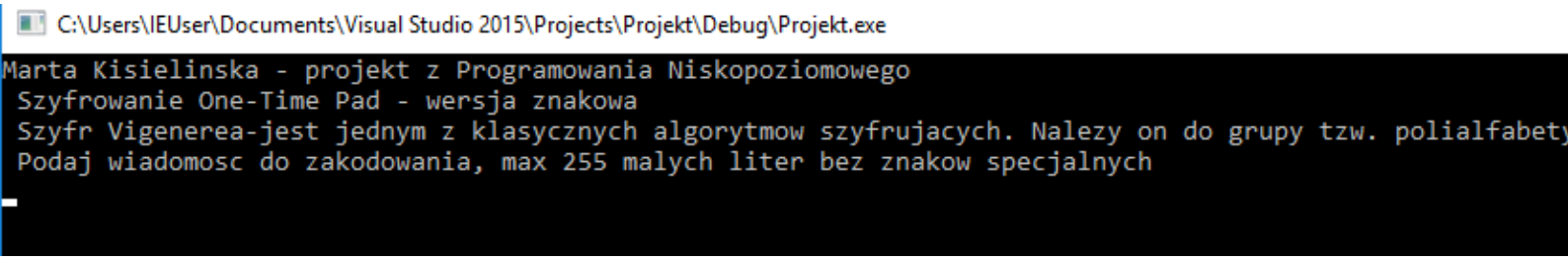
8.odejmujemy wartość 7Ah od rejestru EAX, jest to w systemie dziesiętnym wartość 122d, dzięki czemu mamy w rejestrze wartość przesunięcia względem wartości klucza,

9.dodajemy do wartości przesunięcia liczbę 97d -61h- co odpowiada kodowi ASCII pierwszego interesującego nas znaku czyli literki ‘a’. Dzięki temu mamy w rejestrze EAX kod ASCII zakodowanej litery

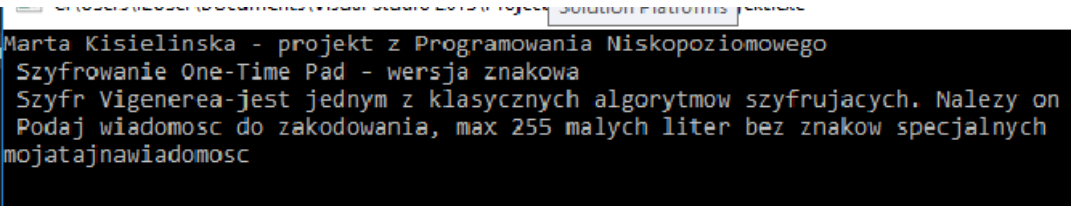
10.przenosimy ją na wolne miejsce w tablicy zakodowane, gdzie mamy gotową zakodowaną wiadomość po ostatnim obiegu pętli.

6. Testowanie

Zacznijmy od początku:

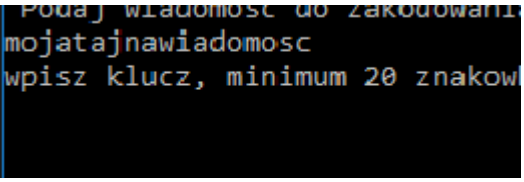


Pierwsze uruchomienie Aplikacji, nazwisko autora, krótki opis i zachęta do wpisania swojej wiadomości do zaszyfrowania razem z ostrzeżeniem o ograniczeniach.



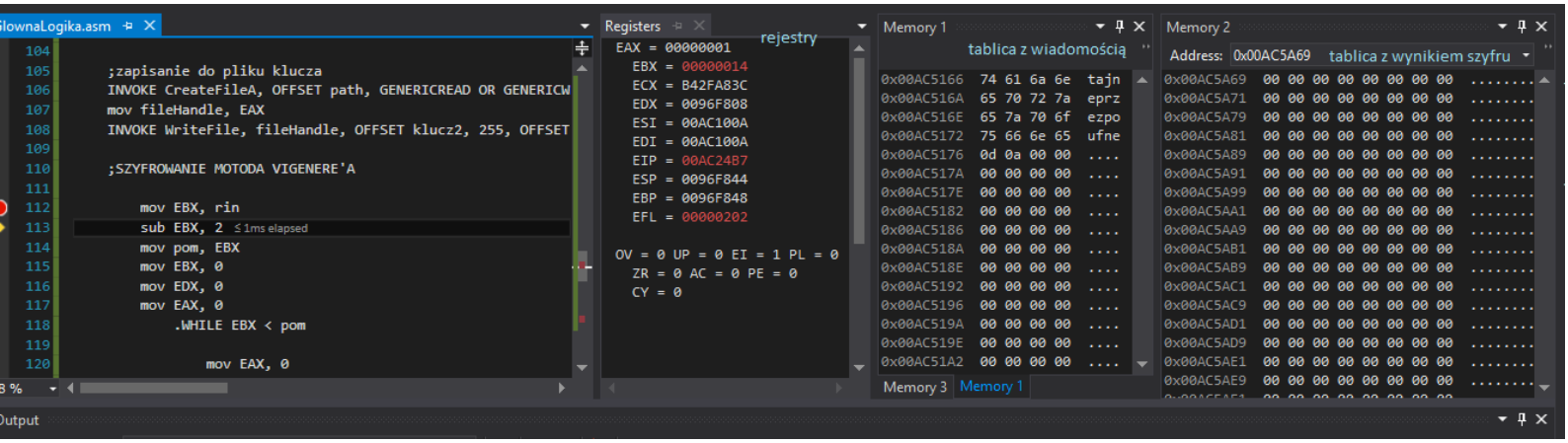
Moja „tajna wiadomość”,

po zatwierdzeniu enterem:

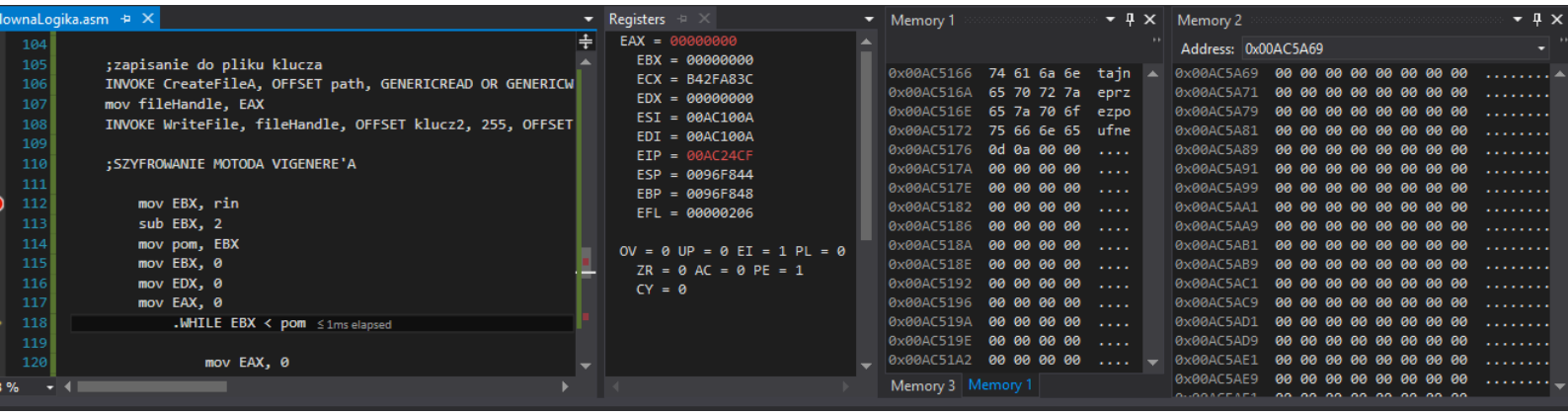


Dzięki procedurze wsprintfA kod „wie” ile muszę wpisać znaków aby wiadomość była odpowiednio zakodowana, użytkownik nie musi liczyć ile znaków podać.

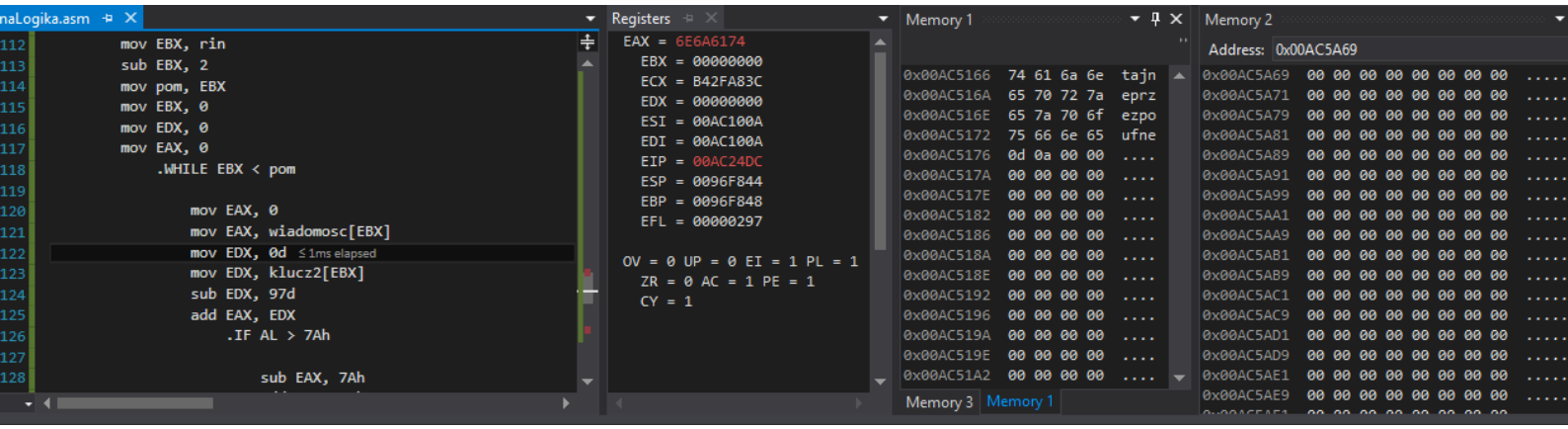
Po wpisaniu „hasła” dzieje się algorytm:



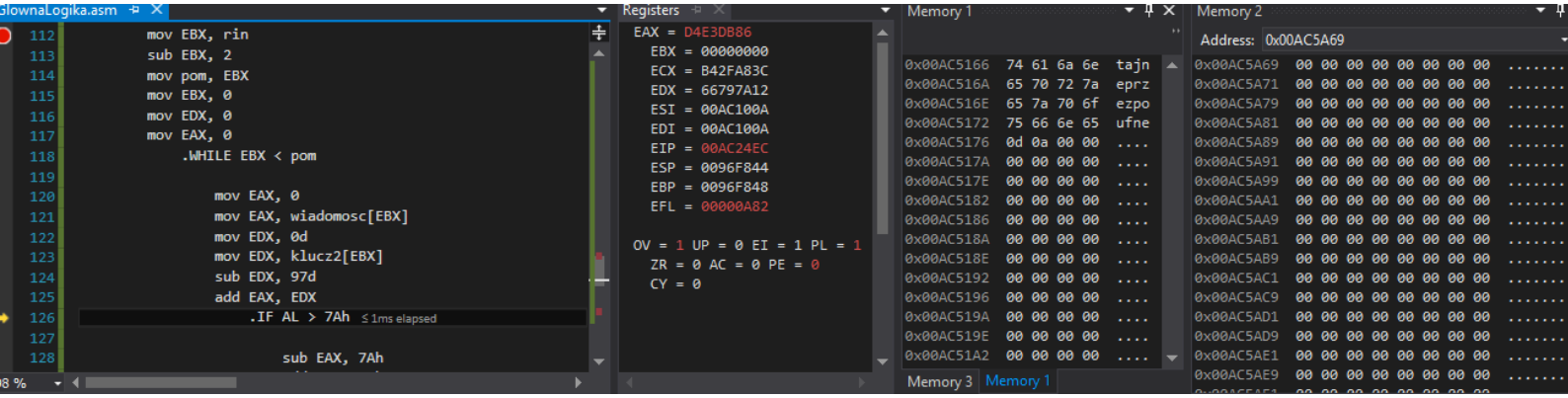
Możemy podejrzeć co dzieje się w rejestrach, przygotowywane są na operowanie danymi z ich pomocą, tj czyszczone.



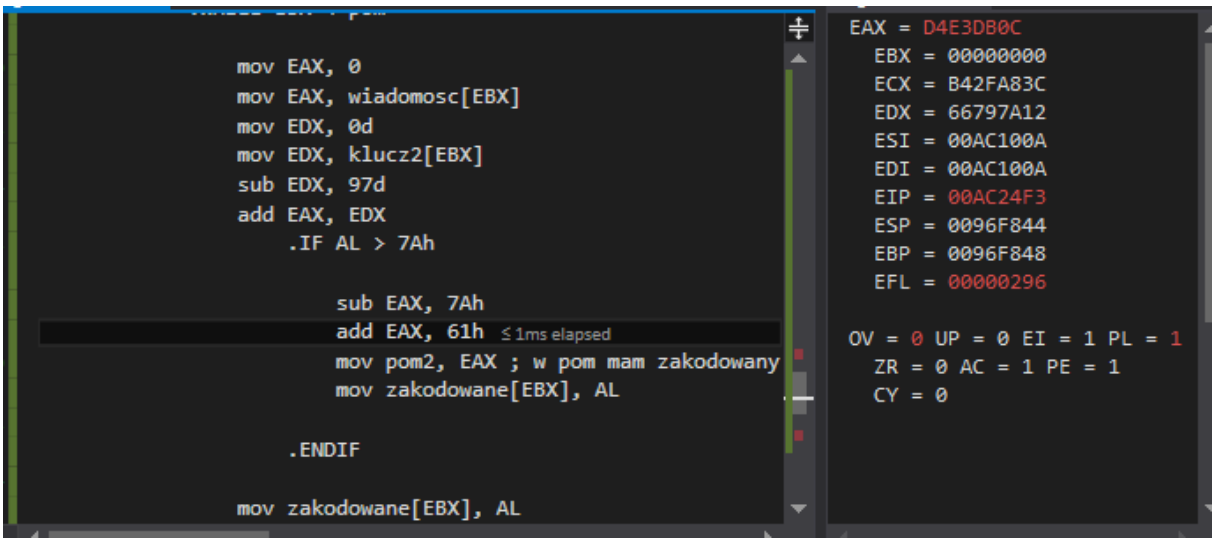
Po przygotowaniu rejestrów możemy przejść do właściwego algorytmu



Algorytm wykonuje się zgodnie z opisem powyżej, krok po kroku, mamy wgląd we wszystko co dzieje się w pamięci i rejestrach

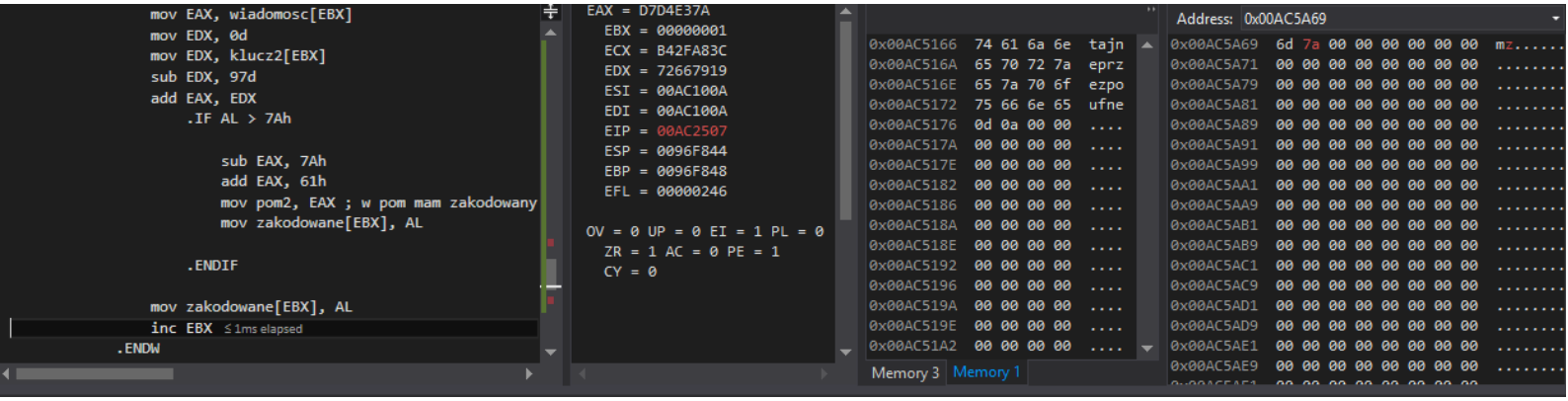


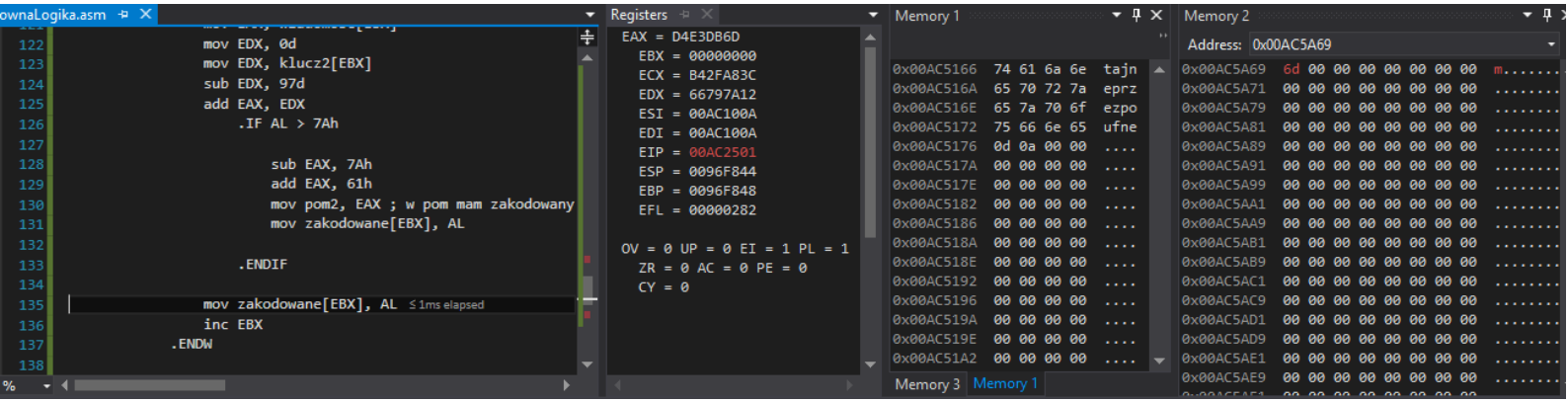
Widzimy tutaj, że wartość sumy kodów ASCII przekroczy 122d, więc wykonujemy instrukcję .IF



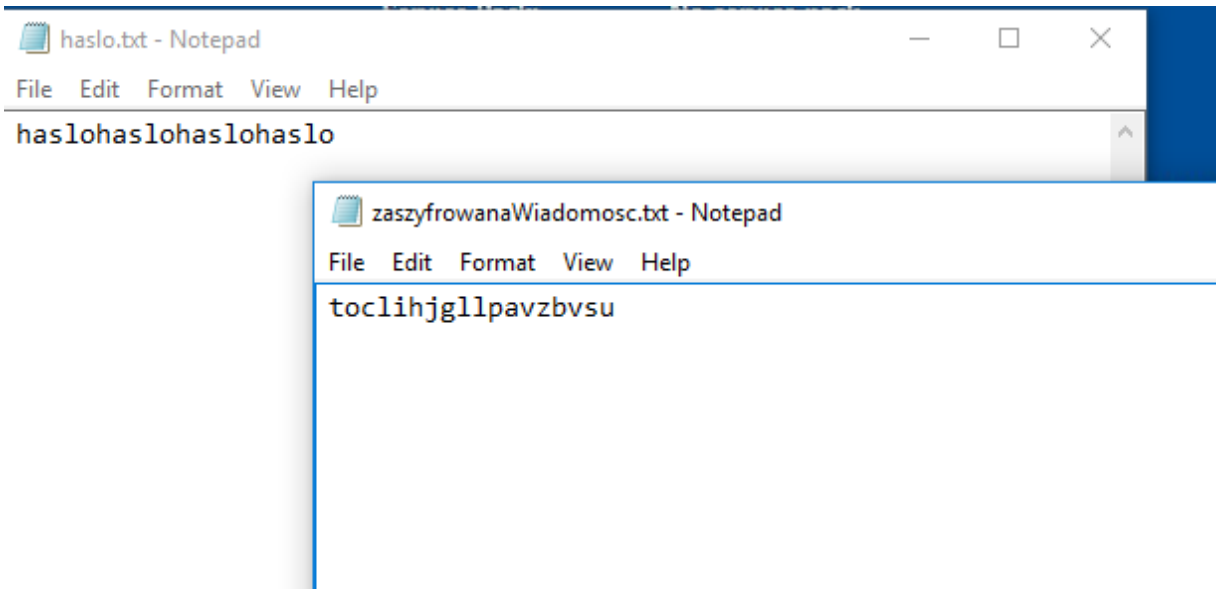
W instrukcji .IF odejmujemy od rejestru EAX 122d aby otrzymać przesunięcie względem pierwszego znaku w alfabecie, potem dodajmy 97d i otrzymujemy w rejestrze EAX kod ASCII odpowiadający zakodowanej już literce

Mamy zakodowany pierwszy znak zgodnie z wcześniejszymi założeniami





Kolejny obieg pętli- kolejny znak w naszej tablicy wynikowej



Na końcu dostajemy dwa pliki – jeden z kluczem, drugi z zaszyfrowaną wiadomością.

7. Podsumowanie

Udało się zrealizować wszystkie założenia projektu biorąc pod uwagę ograniczenia nałożone przy wstępnym projektowaniu funkcjonalności aplikacji.

Możliwe dalsze modyfikacje:
-dodanie modułu losowania klucza za pomocą ziarna, dołączając procedurę losującą, w praktyce taki kod:

```
mov EAX, rin
          sub EAX, 2
          INVOKE wsprintfA, OFFSET solutionBuffer, OFFSET solutionText, EAX
          add ESP, 12
          mov rinp, EAX

          mov EBX, rin
                    sub EBX, 2
                    mov pom3, EBX
                    mov EBX, 0

          ;moduł losowania

          .WHILE EBX < pom3 ; losowanie tyle liter ile ma wiadomość do zakodowania

                              call GetTickCount
                              push EAX
                              call nseed
                              push zakres
                              call nrandom
                              mov wylosowanaLiczba, EAX
                              add EAX, 97
                              mov klucz[EBX], EAX

                              INC EBX
          .ENDW
```

Wraz ze zmiennymi i procedurami:

```
GetTickCount PROTO ;random
nseed PROTO :DWORD ;random
nrandom PROTO :DWORD ;random
.data
  solutionText     BYTE "Wylosuje klucz, minimum %i znaków", 0Ah, 0Dh ,0

  rozmiar4 BYTE $-textWybor
  wybor DD 0
  zakres DWORD 25
  wylosowanaLiczba DWORD 100
  rozmiar3 DD $ - solutionText
  solutionBuffer BYTE 255 dup ( 0 )
```

-dodanie procedury pozwalającej zaszyfrować duże litery i znaki specjalne,
-zapisywanie wiadomości w formacie binarnym, trudniejszym dla odczytu
-graficzny interfejs użytkownika, który jest bardziej estetyczny przy spełnianiu funkcjonalności aplikacji,
-dodanie modułu pozwalającego wybrać metodę szyfrowania np. metodę XOR czy innego szyfrowania.