**תיק פרוייקט**

**שם הפרוייקט:DES (DATA ENCRYPTION STANDART)**

**שם המתכנת: נוה שמחי**

**ת.ז.: 322350224**

**מורה מלווה: פישל נייקלובסקי דוד וייסמן**

**כיתה: י'9**

**בית ספר: הדסים**

**שנת לימודים: תשע"ו 2015-2016**

# תוכן עניינים

תוכן

[תוכן עניינים 2](#_Toc453080181)

[מבוא 3](#_Toc453080182)

[נושא העבודה 4](#_Toc453080183)

[אופן ההפעלה 5](#_Toc453080184)

[גרסאות המערכת 6](#_Toc453080185)

[תיעוד והסבר הפתרון 7](#_Toc453080186)

[תרשימי זרימה 8](#_Toc453080187)

[פעולות 12](#_Toc453080188)

[קוד התוכנית 16](#_Toc453080189)

[דוג' הרצה 81](#_Toc453080190)

[סיכום אישי 83](#_Toc453080191)

# מבוא

**שם העבודה:** הצפנת DES  
**שם הקובץ:** finalproject

**קבצים נלווים:** אין

**סביבת העבודה:** Torbo assembler

**סביבת פיתוח:** notepad++

**סביבת הרצה:** DOSBOX

# נושא העבודה

נושא העבודה שלי הוא הצפנת DES, הצפנה זו פותחה על ידי IBM בשנת 1975,משנת 1976 השתמשו בה להצפנת נתונים בעולם האזרחי עד שנת 2001.

# אופן ההפעלה

בתחילת התוכנית המשתמש יצטרך לבחור מה ירצה לעשות, להצפין או לפענח, אחרי שיבחר הוא יצטרך לבחור איך להכניס את הטקסט ואת המפתח הוא יכול להכניס ערכים לטקסט בשלוש צורות: מספרים אקסדצימליים, אותיות רגילות או לקרוא מקובץ את התוצאה האחרונה שהייתה בתוכנית, למפתח הוא יכול להכניס מספרים אקסדצימליים או אותיות רגילות.

המשתמש יכול להכניס רק 8 מספרים אקדצימליים או 8 אותיות.

# גרסאות המערכת

הגרסה הראשונה: הצפנה מלאה ופענוח מלא שתקבל כקלט מחרוזת תווים של 8 תווים ותדפיס את התוצאה במלואה ומיד אחר כך תחשב ותדפיס את הפענוח.

הגרסה השנייה: רק הצפנה בלי פענוח והודעות בצבעים שונים.

הגרסה השלישית: הצפנה ופענוח והודעות בצבע שחור.

הגרסה הרביעית: הצפנה ופענוח והודעות בצבע שחור.

הגרסה החמישית: הצפנה ופענוח, הודעות בצבע שחור, עם שמירת התוצאה בקובץ וקריאה ממנו.

# תיעוד והסבר הפתרון

התכנית מתחילה בכך שהטקסט עובר IP והמפתח עובר באותו זמן PC1 הטקסט נשאר באותו מספר ביטים אך המספר ביטים של המפתח קטן בבית אחד, בPC1 כל ביט שמיני נשמט וכיוון שיש 8 בתים בהתחלה אז נשמטים 8 ביטים,1 בית.

בכל סיבוב המפתח עובר הזזה מחזורית שמאלה ולאחר מכן עובר PC2, בסיבובים מספר 1,2,9,16 המפתח עובר הזזה מחזורית שמאלה פעם אחת ובכל שאר הפעמים הוא עובר פעמיים ורק לאחר מכן PC2 ומוקטן מ56 ביטים ל48 ביטים, בנוסף באותו סיבוב הצד הימני של הטקסט והמפתח באותו סיבוב עוברים F Function שבהתחלה של הפונקציה הזאת הצד הימני של הטקסט באותו הסיבוב עובר תיבת הרחבה שהוא מורחב מ32 ביטים ל48 ביטים ומבצע את פעולת הXOR עם המפתח של אותו סיבוב, התוצאה עוברת ב8 SBOX שכל אחת לוקחת 6 ביטים וממירה ל4, התוצאה שיוצאת עוברת תיבת P שמשאירה את התוצאה ב32 ביטים אך משנה את סדרם.

כשלמדתי את ההצפנה נעזרתי בשתי הרצאות של Christof paar

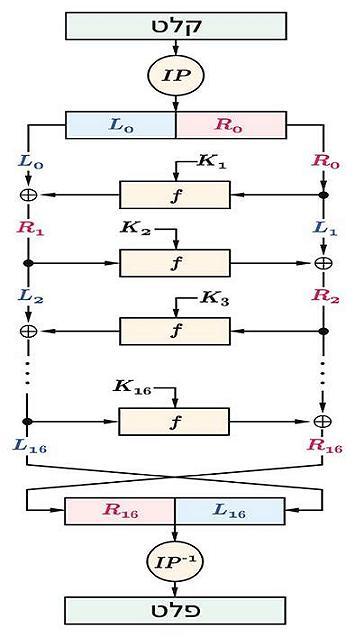
הרצאה מס' 1:

<https://www.youtube.com/watch?v=kPBJIhpcZgE>

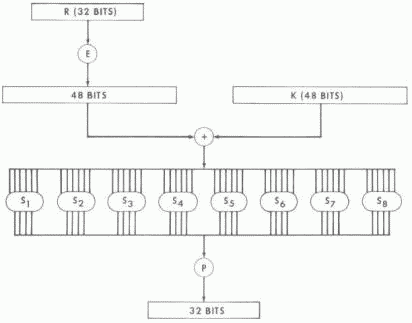
הרצאה מס' 2:

<https://www.youtube.com/watch?v=l-7YW06BFNs>

# https://i.gyazo.com/c597e5389c1954f87125481a1f1e11e4.pngתרשימי זרימה

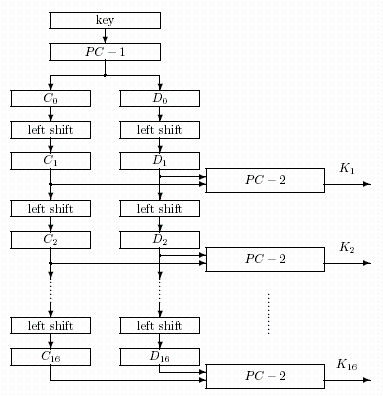


לאחר הקליטה מחלקים את הטקסט לשני חלקים, ימני ושמאלי, בכל סיבוב הצד הימני של אותו הסיבוב ומעביר אותו בפונ' F ולאחר מכן התוצאה עוברת XOR עם הצד השמאלי של אותו הסיבוב והצדדים מתחלפים, הצד השמאלי של הסיבוב הבא יהיה שווה לצד הימני של הסיבוב הנוכחי והצד הימני של הסיבוב הבא יהיה שווה לתוצאת XOR של הצד השמאלי של אותו סיבוב והתוצאה שיצאה מהפונ' F.

**f Function**

הפונ' הזו מבצעת את תיבת ההרחבה E שמרחיבה את הצד הימני של הטקסט מ32 ביטים ל48 ביטים, באותו זמן המפתח עובר את הKEY SCHEDULE שזה מה שהמפתח עובר בסיבוב אחד, ולאחר מכן עושה XOR בין שתי התוצאות, מעביר את התוצאה של הXOR בSBOX בסופו התוצאה מצטמצמת ל32 ביטים והם עוברים בתיבת תמורה P ששומרת על הגודל של התוצאה.

**key schedule**



בהתחלה המפתח עובר בטבלה הסטטית PC1 ואז הפונ' בודקת אם מה המספר סיבוב ואם הסיבוב הוא מס' 1,2,8,16 אז המפתח יעבור הזזה מחזורית לשמאל פעם אחת אחרת, הוא יעבור פעמיים הזזה מחזורית שמאלה, לאחר ההזזה המפתח עובר בטבלה סטטית PC2.

# פעולות

1. NewLine -הפעולה לא מקבלת פרמטרים ותפקידה היא להוריד שורה בחלון המשתמש וחוזר לתחילת השורה. \*פרוצדורה\*.
2. PushAll – פעולה הדוחפת 6 רגיסטרים למחסנית ax,bx,cx,dx,di,si . \*מקרו\*.
3. OpenFile – פותח קובץ ששמו כתוב במשתנה FileName ומקובץ זה נקרא את הטקסט הרשום בו או נכתוב אליו, הפרמטרים שלה נמצאים במשתנים, פרמטר אחד הוא שם הקובץ, פרמטר שני הוא ה FileHandle של הקובץ שקיבלנו בהתחלה, הפעולה לא מחזירה כלום, אלא אם לא תהיה אפשרות לוח קובץ ואז בAX יהיה מספר מתאים. \*פרוצדורה\*.
4. WriteToFile –הפעולה מעתיקה את מה שיש במשתנה encriptionordecryptionresult לקובץ ששמו נמצא במשתנה FileName \*פרוצדורה\*.
5. CreateFile- יוצר קובץ בשם FILENAME,הפעולה מקבלת כפרמטרים את שם הקובץ וה FileHandle. \*פרוצדורה\*.
6. DeleteFile – מוחק את הקובץ ששמו נמצא ב FILENAME \*פרוצדורה\*.
7. ReadFromFile – קורא מקובץ ושם את מה שכתוב שם ב hextext, הפעולה מקבלת כפרמטר את הFILEHANDLE ומעתיקה 8 בתים. \*פרוצדורה\*.
8. PopAll– פעולה המוציאה 6 רגיסטרים מהמחסנית ax,bx,cx,dx,di,si \*מקרו\*.
9. hextextkelet- כשהטקסט נקלט באקסדצימלי בזיכרון הוא כתוב בקודי הסקי אז מקרו זה גורם להם להיות במספרים במקום קוד הסקי אין פרמטרים והפעולה מחזירה את המספרים האקסדצימליים\*מקרו\*.
10. hexkeykelet- כשהמפתח נקלט באקסדצימלי בזיכרון הוא כתוב בקודי הסקי אז מקרו זה גורם להם להיות במספרים במקום קוד הסקי מקרו זו לא מקבלת פרמטרים\*מקרו\*.
11. RemoveEntertext– מוריד את האנטר מהקלט של הטקסט \*פרוצדורה\*.
12. RemoveEnterkey- מוריד את האנטר מהקלט של המפתח אין פרמטרים מועברים לפרוצודרה והפרוצדורה מחזירה את הטקסט ללא קוד הסקי של אנטר, אין ערכים מועברים לפעולה \*פרוצדורה\*.
13. ChangeRandL- מחליף בין הצד הימני של הטקסט לשמאלי אין פרמטרים \*מקרו\*.
14. keyoner – מבצע את מה שהמפתח עובר בסיבוב אחד אין פרמטרים \*פרוצדורה\*.
15. Ffunction1- חצי מהF FUNCTION של ההצפנה, מבצע את תיבת ההרחבה E עם הפרמטר של הצד הימני של הטקסט באותו סיבוב, קורא לפעולה keyoner מבצע את הXOR בין שתי התוצאות, הראשונה של ה keyoner והשנייה היא התוצאה של תיבת ההרחבה P ומעביר את התוצאה של הXOR ב4 מתוך 8 תיבות SBOX. \*מקרו\*.
16. Ffunction2- חצי שני מהF FUNCTION של ההצפנה, מעביר את התוצאה של הXOR ב4 תיבות SBOX שנותרו ואז מעביר בתיבת תמורה P ועושה XOR של התוצאה מתיבת התמורה עם הצד השמאלי של הטקסט באותו סיבוב. \*מקרו\*.
17. Ffunction1D- חצי מהF FUNCTION של הפענוח, מבצע את תיבת ההרחבה E עם הפרמטר של הצד הימני של הטקסט באותו סיבוב, קורא לפעולה keyoner (ההבדל בין ההצפנה לפענוח הוא שבפענוח בסיבוב הראשון הוא קורא לפעולה 16 פעמים כי זה הסיבוב בהצפנה השקול לסיבוב זה בפיענוח סיבוב i בפענוח שקול לסיבוב 16-i בהצפנה), מבצע את הXOR בין שתי התוצאות, הראשונה של ה keyoner והשנייה היא התוצאה של תיבת ההרחבה P ומעביר את התוצאה של הXOR ב4 מתוך 8 תיבות SBOX. \*מקרו\*.
18. Ffunction2D- חצי שני מהF FUNCTION של הפענוח, מעביר את התוצאה של הXOR ב4 תיבות SBOX שנותרו ואז מעביר בתיבת תמורה P ועושה XOR של התוצאה מתיבת התמורה עם הצד השמאלי של הטקסט באותו סיבוב. \*מקרו\*.
19. Rotatecall- מקרו שקורא לפרוצדורה שמזיזה הזזה מחזורית שמאלה את המפתח ומעבירה את הערכים לשני משתנים. \*מקרו\*.
20. getBit – לדחוף אופסט ומס' ביט ולקבל את הביט במקום הזה הפרמטרים הם, אופסט של הטקסט שממנו רוצים לקחת את הביט ואחריו דוחפים את מספר הביט(אם יש 64 ביטים אזי צריכים לדחוץ מספר בין 0-63 שהוא המספר הסידורי של הביט בטקסט), לאחר קריאת הפרוצדורה צריך לעשות POP לאחד מהרגיסטרים ששם יהיה את הביט או 0 או 1\*פרוצדורה\*.
21. MainPc1orrotate – משמש לעשות את הפעולה PC1 וRotate \*פרוצדורה\* צריך לדחוף למחסנית את האופסט של המשתנה שבו תהיה שמורה התוצאה ולאחר מכן דוחפים את האופסט של הטקסט שעובר את הפעולה ואחריו דוחפים את האופסטים של הטבלאות של אחת מהפעולות אחד אחרי השני, הפעולה PC1 היא פעולה אשר מורידה כל ביט שמיני מהמפתח, הפעולה ROTATE כמו שכתוב למעלה, מזיזה את כל הביטים בהזזה מחזורית שמאלה.
22. MainExpansion- מבצע את הפעולה Expansion דוחפים את האופסט של המשתנה שאנו רוצים שתהיה בו התוצאה של הפעולה ואחריו את האופסט של המשתנה שיש בו את הטקסט לפני הפעולה \*פרוצדורה\*.
23. sbox1 – מעביר 6 ביטים בטבלה sbox1. אין פרמטרים מועברים \*פרוצדורה\*
24. Sbox2 – מעביר 6 ביטים בטבלה sbox2. אין פרמטרים מועברים \*פרוצדורה\*
25. sbox3 – מעביר 6 ביטים בטבלה sbox3.אין פרמטרים מועברים \*פרוצדורה\*
26. sbox4 – מעביר 6 ביטים בטבלה sbox4. אין פרמטרים מועברים \*פרוצדורה\*
27. sbox5 – מעביר 6 ביטים בטבלה sbox5. אין פרמטרים מועברים \*פרוצדורה\*
28. sbox6 – מעביר 6 ביטים בטבלה sbox6. אין פרמטרים מועברים \*פרוצדורה\*
29. sbox7 – מעביר 6 ביטים בטבלה sbox7. אין פרמטרים מועברים \*פרוצדורה\*
30. sbox8 – מעביר 6 ביטים בטבלה sbox8. אין פרמטרים מועברים \*פרוצדורה\*
31. MainIp- מעביר את הטקסט בטבלה IP או IP-1 הפרמטרים הם האופסט של המשתנה שאנו רוצים שתהיה בו התוצאה של הפעולה ואחריו את האופסט של המשתנה שיש בו את הטקסט לפני הפעולה, לאחר מכן מעבירים 8 אופסטים, של הטבלאות של הפעולה שנקראת IP או הטבלאות של הפעולה IP-1. \*פרוצדורה\*
32. MainPc2- מעביר את המפתח בכל סיבוב בטבלה PC2 אחרי שהמפתח עבר ROTATE הפרמטרים לפי הסדר: אופסט של משתנה שבו תהיה שמורה התוצאה של אחרי הפעולה, אופסט של המשתנה שאנו מבצעים עליו את הפעולה \*פרוצדורה\*
33. MainPer – מעביר את התוצאה מהSBOX בטבלה שנקראת PERMATION. תחילה מעבירים את האופסט של המשתנה שאנו רוצים שתהיה בו התוצאה של הפעולה ואחריו את האופסט של המשתנה שיש בו את הטקסט לפני הפעולה \*פרוצדורה\*
34. ENCRYPTION1 - כל ההצפנה אין פרמטרים, חייב לעשות mov sp,100h הפעולה לא מקבלת פרמטרים אך משתמשת בכל הפעולות המוזכרות. \*פרוצדורה\*
35. Decryption1- כל הפענוח אין פרמטרים, חייב לעשות mov sp,100h, הפעולה לא מקבלת פרמטרים אך היא משתמשת בכל הפעולות המוזכרות. \*פרוצדורה\*
36. Keyandtextstartscreen – הקלט והמסך בהתחלה אין פרמטרים והפעולה מחזירה את הקלט המסודר, לדוגמא אם נקלט קלט מסויים בהקס דצימלי המחשב לא יכתוב את הצורה שבה הוא כתוב בתוך הזיכרון ואז ההצפנה לא תעבוד על המפתח שרצינו לכן אנו ממירים את הקלט מקודי האסקי שלו למספרים הקסדצימליים. \*פרוצדורה\*

# קוד התוכנית

IDEAL

MODEL small

STACK 100h

DATASEG

; --------------------------

; Your variables here

des02 db' /\ \ /\ \ /\ \ ',13,10,'$'

des03 db' /::\ \ /::\ \ /::\ \ ',13,10,'$'

des04 db' /::::\ \ /::::\ \ /::::\ \ ',13,10,'$'

des05 db' /::::::\ \ /::::::\ \ /::::::\ \ ',13,10,'$'

des06 db' /:::/\:::\ \ /:::/\:::\ \ /:::/\:::\ \ ',13,10,'$'

des07 db' /:::/ \:::\ \ /:::/\_\_\:::\ \ /:::/\_\_\:::\ \ ',13,10,'$'

des08 db' /:::/ \:::\ \ /::::\ \:::\ \ \:::\ \:::\ \ ',13,10,'$'

des09 db' /:::/ / \:::\ \ /::::::\ \:::\ \ \_\_\_\:::\ \:::\ \ ',13,10,'$'

des10 db' /:::/ / \:::\ \_\_\_\ /:::/\:::\ \:::\ \ /\ \:::\ \:::\ \ ',13,10,'$'

des11 db'/:::/\_\_\_\_/ \:::| |/:::/\_\_\:::\ \:::\\_\_\_\_\/::\ \:::\ \:::\\_\_\_\_\',13,10,'$'

des12 db'\:::\ \ /:::|\_\_\_\_|\:::\ \:::\ \::/ /\:::\ \:::\ \::/ /',13,10,'$'

des13 db' \:::\ \ /:::/ / \:::\ \:::\ \/\_\_\_\_/ \:::\ \:::\ \/\_\_\_\_/ ',13,10,'$'

des14 db' \:::\ \ /:::/ / \:::\ \:::\ \ \:::\ \:::\ \ ',13,10,'$'

des15 db' \:::\ /:::/ / \:::\ \:::\\_\_\_\_\ \:::\ \:::\\_\_\_\_\ ',13,10,'$'

des16 db' \:::\ /:::/ / \:::\ \::/ / \:::\ /:::/ / ',13,10,'$'

des17 db' \:::\/:::/ / \:::\ \/\_\_\_\_/ \:::\/:::/ / ',13,10,'$'

des18 db' \::::::/ / \:::\ \ \::::::/ / ',13,10,'$'

des19 db' \::::/ / \:::\\_\_\_\_\ \::::/ / ',13,10,'$'

des20 db' \::/\_\_\_\_/ \::/ / \::/ / ',13,10,'$'

des21 db' ~~ \/\_\_\_\_/ \/\_\_\_\_/ ',13,10,'$'

Rotate0 db 1,2,3,4,5,6,7,8

Rotate1 db 9,10,11,12,13,14,15,16

Rotate2 db 17,18,19,20,21,22,23,24

Rotate3 db 25,26,27,0,29,30,31,32

Rotate4 db 33,34,35,36,37,38,39,40

Rotate5 db 41,42,43,44,45,46,47,48

Rotate6 db 49,50,51,52,53,54,55,28

Subtation10 db 14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7

Subtation11 db 0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8

Subtation12 db 4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0

Subtation13 db 15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13

Subtation20 db 15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10

Subtation21 db 3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5

Subtation22 db 0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15

Subtation23 db 13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9

Subtation30 db 10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8

Subtation31 db 13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1

Subtation32 db 13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7

Subtation33 db 1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12

Subtation40 db 7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15

Subtation41 db 13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9

Subtation42 db 10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4

Subtation43 db 3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14

Subtation50 db 2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9

Subtation51 db 14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6

Subtation52 db 4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14

Subtation53 db 11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3

Subtation60 db 12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11

Subtation61 db 10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8

Subtation62 db 9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6

Subtation63 db 4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13

Subtation70 db 4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1

Subtation71 db 13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6

Subtation72 db 1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2

Subtation73 db 6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12

Subtation80 db 13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7

Subtation81 db 1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2

Subtation82 db 7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8

Subtation83 db 2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11

erormes db 'this is not hex numbers',13,10,'$'

mes0 db 'what would you like to do? 1 for ENCRYPTION 2 for DECRYPTION',13,10,'$'

mes1 db 'would you like to enter the text at hex(1) or normal letters (2) or Read from file(3)?',13,10,'$'

mes2 db 'would you like to enter the key at hex(1) or normal letters (2)?',13,10,'$'

mes3 db 'you did not type 1 ,2 or 3',13,10,'$'

mes8 db 'you did not type 1 ,2',13,10,'$'

mes4 db 'enter the text',13,10,'$'

mes5 db 'enter the key',13,10,'$'

mes6 db "Notice:You can't read form the file$"

mes7 db "Error: couldn't create new text file$"

encrypres db 'the encryption result is: ',13,10,'$'

decrypres db 'the dycription result is: ',13,10,'$'

textafterip db 8 dup (?)

plaintextkelet db 11 dup (?)

plainkeykelet db 11 dup (?)

textbefore db 8 dup (?)

Rightsidetext db 4 dup (?)

Leftsidetext db 4 dup (?)

afterEx db 6 dup (?)

KeyAfterPC1Beforepc2beforerotateandmain db 7 dup (?)

KeyDecTemp db 7 dup (?)

afterro db 7 dup (?)

keyAfterpc2 db 6 dup (?)

aftersbox db 4 dup (?)

afterper db 4 dup (?)

hexkey db 8 dup (?);key that I can work on him without the enter letter code/

hextext db 8 dup (?) ;text that I can work on him without the enter letter code/

encriptionordecryptionresult db 8 dup (?), '$'

ip0 db 57,49,41,33,25,17,9,1

ip1 db 59,51,43,35,27,19,11,3

ip2 db 61,53,45,37,29,21,13,5

ip3 db 63,55,47,39,31,23,15,7

ip4 db 56,48,40,32,24,16,8,0

ip5 db 58,50,42,34,26,18,10,2

ip6 db 60,52,44,36,28,20,12,4

ip7 db 62,54,46,38,30,22,14,6

ip\_10 db 39,7,47,15,55,23,63,31

ip\_11 db 38,6,46,14,54,22,62,30

ip\_12 db 37,5,45,13,53,21,61,29

ip\_13 db 36,4,44,12,52,20,60,28

ip\_14 db 35,3,43,11,51,19,59,27

ip\_15 db 34,2,42,10,50,18,58,26

ip\_16 db 33,1,41,9,49,17,57,25

ip\_17 db 32,0,40,8,48,16,56,24

Expansion0 db 31,0,1,2,3,4,3,4

Expansion1 db 5,6,7,8,7,8,9,10

Expansion2 db 11,12,11,12,13,14,15,16

Expansion3 db 15,16,17,18,19,20,19,20

Expansion4 db 21,22,23,24,23,24,25,26

Expansion5 db 27,28,27,28,29,30,31,0

PC10 db 56,48,40,32,24,16,8,0

PC11 db 57,49,41,33,25,17,9,1

PC12 db 58,50,42,34,26,18,10,2

PC13 db 59,51,43,35,62,54,46,38

PC14 db 30,22,14,6,61,53,45,37

PC15 db 29,21,13,5,60,52,44,36

PC16 db 28,20,12,4,27,19,11,3

PC20 db 13,16,10,23,0,4,2,27

PC21 db 14,5,20,9,22,18,11,3

PC22 db 25,7,15,6,26,19,12,1

PC23 db 40,51,30,36,46,54,29,39

PC24 db 50,44,32,47,43,48,38,55

PC25 db 33,52,45,41,49,35,28,31

Permutation0 db 15,6,19,20,28,11,27,16

Permutation1 db 0,14,22,25,4,17,30,9

Permutation2 db 1,7,23,13,31,26,2,8

Permutation3 db 18,12,29,5,21,10,3,24

Encryptionresult db 8 dup (?)

Decryptionresult db 8 dup (?)

NumberOfBytesToRead db 6 dup (?)

FileHandle dw ?

FileName db "naveproject.txt"

; --------------------------

CODESEG

CODESEG

proc NewLine

push ax

push dx

mov dl, 0Dh

mov ah, 2h

int 21h

mov dl, 0Ah

mov ah, 2h

int 21h

pop dx

pop ax

ret

endp

;=====================================

;push the registers ax,bx,cx,dx,di,si|

;=====================================

macro PushAll; pushes all registers

push ax

push bx

push cx

push dx

push di

push si

endm PushAll

;==============================================================================================

;open a text file that the encrypted or decrypted text will be put on or to read from the file|

;==============================================================================================

proc OpenFile

mov ah, 3Dh

mov al,2

mov dx,offset FileName

int 21h

jc openerror

mov [word ptr FileHandle],ax

ret

openerror:

mov dx, offset mes6

mov ah, 9h

int 21h

call NewLine

ret

endp OpenFile

;===============================

;writes the text on the file |

;the parameters are file handle|

;===============================

proc WriteToFile

mov ah,40h

mov bx,[word ptr FileHandle]

mov cx,8

mov dx,offset encriptionordecryptionresult

int 21h

ret

endp WriteToFile

;=============================================

;create file |

;the parameters are file handle and file name|

;=============================================

proc CreateFile

mov ah,3ch

mov dx,offset FileName

xor cx,cx

int 21h

jc ErrorCreate

mov [word ptr FileHandle],ax

ret

ErrorCreate:

mov dx,offset mes7

mov ah,9h

int 21h

ret

endp CreateFile

;=============================================

;Delete file |

;the parameters are file handle and file name|

;=============================================

proc DeleteFile

mov dx,offset FileName

mov ah,41h

int 21h

ret

endp DeleteFile

;=====================================================================

;read the text in the file and copy the text to the variable hextext|

;the parameters are file handle and file name |

;=====================================================================

proc ReadFromFile

mov ah,3Fh

mov bx,[word ptr FileHandle]

mov cx,8

mov dx,offset hextext

int 21h

jc ErrorMessage

ret

ErrorMessage:

call NewLine

mov dx,offset mes6

mov ah,9h

int 21h

jmp exit

endp ReadFromFile

;====================================

;pop the registers ax,bx,cx,dx,di,si|

;====================================

macro PopAll

pop si

pop di

pop dx

pop cx

pop bx

pop ax

endm PopAll

;=====================================================================

;makes the text to be in hexdecimal and not int ascii codes |

;no parameters |

;=====================================================================

macro hextextkelet

mov bx, offset hextext

xor cx,cx

turntohex:

mov al, [byte ptr bx]

cmp al,0

je hexlet

cmp al, '0'

jb Erortype

cmp al, '9'

ja letter

sub al,'0'

jmp hexlet

letter:

cmp al,'a'

jb Erortype

cmp al,'f'

ja Erortype

sub al, 'a'

add al,10

hexlet:

mov [byte ptr bx],al

inc bx

inc cx

cmp cx, 8

jne turntohex

jmp endmacrowithouttypeeror

Erortype:

mov dx, offset erormes

mov ah,9

int 21h

jmp exit

call NewLine

endmacrowithouttypeeror:

endm hextextkelet

;=====================================================================

;makes the key to be in hexdecimal and not int ascii codes |

;no parameters |

;=====================================================================

macro hexkeykelet

mov bx, offset hexkey

xor cx,cx

turntohexK:

mov al, [byte ptr bx]

cmp al,0

je hexletK

cmp al, '0'

jb ErortypeK

cmp al, '9'

ja letterK

sub al,'0'

jmp hexletK

letterK:

cmp al,'a'

jb ErortypeK

cmp al,'f'

ja ErortypeK

sub al, 'a'

add al,10

hexletK:

mov [byte ptr bx],al

inc bx

inc cx

cmp cx, 8

jne turntohexK

jmp Endmes

ErortypeK:

mov dx, offset erormes

mov ah,9

int 21h

call NewLine

jmp exit

Endmes:

endm hexkeykelet

;=============================================================================================================

;remove the Enter code from the text and copy the text from plaintextkelet to hextext to pass the buffer codes|

;no parameters |

;=============================================================================================================

proc RemoveEntertext

xor ax,ax

xor cx,cx

mov bx,offset plaintextkelet

mov di,offset hextext

inc bx

inc bx

Labbdikaofenter:

mov ax, [word ptr bx]

cmp ah, 13

jne labnoenterfi

mov ah,0

jmp labnoenter

labnoenterfi:

cmp al, 13

jne labnoenter

mov al,0

labnoenter:

mov [word ptr di], ax

inc bx

inc bx

inc di

inc di

inc cx

cmp cx, 4

jne Labbdikaofenter

ret

endp

;===========================================================================================================

;remove the Enter code from the key and copy the text from plainkeykelet to hexkey to pass the buffer codes|

;no parameters |

;===========================================================================================================

proc RemoveEnterkey

xor ax,ax

xor cx,cx

mov bx,offset plainkeykelet

mov di,offset hexkey

inc bx

inc bx

LabbdikaofenterK:

mov ax, [word ptr bx]

cmp ah, 13

jne labnoenterfirK

mov ah,0

labnoenterfirK:

cmp al, 13

jne labnoenterK

mov al,0

labnoenterK:

mov [word ptr di], ax

inc bx

inc bx

inc di

inc di

inc cx

cmp cx, 4

jne LabbdikaofenterK

ret

endp

;======================================================

;change the right side and the left side of the text |

;no parameters uses the Rightsidetext and Leftsidetext| |

;======================================================

macro ChangeRandL

push ax bx

mov ax, [word ptr Rightsidetext]

mov bx, [word ptr Leftsidetext]

mov [word ptr Leftsidetext],ax

mov [word ptr Rightsidetext],bx

mov ax, [word ptr Rightsidetext+2]

mov bx, [word ptr Leftsidetext+2]

mov [word ptr Leftsidetext+2],ax

mov [word ptr Rightsidetext+2],bx

pop bx ax

endm ChangeRandL

;=================================

;does the one round key schedule |

;no parameters |

;=================================

proc keyoner

call Rotatecall

cmp si, 0

je endmacofro

cmp si, 1

je endmacofro

cmp si, 8

je endmacofro

cmp si, 15

je endmacofro

oneroundofrotate:

call Rotatecall

endmacofro:

push offset keyAfterpc2

push offset afterro

call MainPc2

mov dx,ax

pop ax

pop ax

mov ax,dx

ret

endp

;==================================

;one half of f function Encryption|

;no parameters |

;==================================

macro Ffunction1 ;all the one round key

push offset afterEx

push offset Rightsidetext

call MainExpansion

call keyoner

mov ax, [word ptr keyAfterpc2]

xor [word ptr afterEx],ax

mov ax, [word ptr keyAfterpc2+2]

xor [word ptr afterEx+2],ax

mov ax, [word ptr keyAfterpc2+4]

xor [word ptr afterEx+4],ax

push ax

mov ax, [word ptr afterEx]

push ax

call sbox1

mov ax, [word ptr afterEx]

push ax

call sbox2

mov ax, [word ptr afterEx]

push ax

mov ax, [word ptr afterEx+2]

push ax

call sbox3

mov ax, [word ptr afterEx+2]

push ax

call sbox4

endm Ffunction1

;=====================================

;second half of f function Encryption|

;no parameters |

;=====================================

macro Ffunction2

mov ax, [word ptr afterEx+2]

push ax

call sbox5

mov ax, [word ptr afterEx+2]

push ax

mov ax, [word ptr afterEx+4]

push ax

call sbox6

mov ax, [word ptr afterEx+4]

push ax

call sbox7

mov ax, [word ptr afterEx+4]

push ax

call sbox8

push offset afterper

push offset aftersbox

call MainPer

mov ax, [word ptr afterper]

xor [word ptr Leftsidetext],ax

mov ax, [word ptr afterper+2]

xor [word ptr Leftsidetext+2],ax

ChangeRandL

endm Ffunction2

;==================================

;one half of f function Decryption|

;no parameters |

;==================================

macro Ffunction1D ;all the one round key

push si

push offset afterEx

push offset Rightsidetext

call MainExpansion

pop di

pop di

pop si

push si bx

mov bx,16

sub bx,si

mov si,0

labhowmanytimesthekey:

PushAll

call keyoner

PopAll

inc si

cmp bx,si

jne labhowmanytimesthekey

pop bx si

PushAll

mov ax,0

mov bx,offset KeyAfterPC1Beforepc2beforerotateandmain

mov si,offset KeyDecTemp

movkeyDecTemptoKeyafterpc1:

mov cl,[byte ptr si]

inc si

mov [byte ptr bx], cl

inc bx

inc ax

cmp ax, 7

je labhere1Deccon1

jmp movkeyDecTemptoKeyafterpc1

labhere1Deccon1:

PopAll

mov ax, [word ptr keyAfterpc2]

xor [word ptr afterEx],ax

mov ax, [word ptr keyAfterpc2+2]

xor [word ptr afterEx+2],ax

mov ax, [word ptr keyAfterpc2+4]

xor [word ptr afterEx+4],ax

push ax

mov ax, [word ptr afterEx]

push ax

call sbox1

mov ax, [word ptr afterEx]

push ax

call sbox2

mov ax, [word ptr afterEx]

push ax

mov ax, [word ptr afterEx+2]

push ax

call sbox3

mov ax, [word ptr afterEx+2]

push ax

call sbox4

endm Ffunction1D

;=====================================

;second half of f function Decryption|

;no parameters |

;=====================================

macro Ffunction2D

mov ax, [word ptr afterEx+2]

push ax

call sbox5

mov ax, [word ptr afterEx+2]

push ax

mov ax, [word ptr afterEx+4]

push ax

call sbox6

mov ax, [word ptr afterEx+4]

push ax

call sbox7

mov ax, [word ptr afterEx+4]

push ax

call sbox8

push offset afterper

push offset aftersbox

call MainPer

mov ax, [word ptr afterper]

xor [word ptr Leftsidetext],ax

mov ax, [word ptr afterper+2]

xor [word ptr Leftsidetext+2],ax

ChangeRandL

endm Ffunction2D

;=================================================================================================================

;one call to left rotate and copy the result to two variables afterro and KeyAfterPC1Beforepc2beforerotateandmain|

;no parameters |

;=================================================================================================================

proc Rotatecall

push offset afterro

push offset KeyAfterPC1Beforepc2beforerotateandmain

push offset Rotate0

push offset Rotate1

push offset Rotate2

push offset Rotate3

push offset Rotate4

push offset Rotate5

push offset Rotate6

call MainPc1orrotate

PushAll

mov ax,0

mov bx,offset KeyAfterPC1Beforepc2beforerotateandmain

mov si,offset afterro

afterrotatetobeforerotate1:

mov cl,[byte ptr si]

inc si

mov [byte ptr bx], cl

inc bx

inc ax

cmp ax, 7

je labhere12

jmp afterrotatetobeforerotate1

labhere12:

PopAll

ret

endp

;===========================================================

;push offset of the text you want the bit from |

;push number of bit at the text |

;call getBit |

;pop Bit it's the bit that you wanted (1/0) |

;===========================================================

PROC getBit

DATAa EQU [bp+14]

INDXa EQU [bp+16]

push AX BX CX DX DI BP

mov bp , sp

mov bx, DATAa

mov ax, INDXa

mov ah,0

mov dl, 8

div dl ;how many bytes to mov from the start of the start ז"א כמה בתים אנו צריכים לעבור כדי להגיע לביט שאנו צריכים נגיד: אם אנו רוצים ביט מס' 8 אז הוא נמצא בבית מס' 2

mov dl,ah ;לשמור את השארית מהחילוק (איזה מס' ביט הוא בבית שהוא נמצא)

xor ah,ah ;לאפס את השארית כדי שנישאר רק עם כמה בתים אנו צריכים להוסיף כדי להגיע לבית שבו נמצא הביט

mov di,ax

add bx,di

mov al, [byte ptr bx]

push cx ;save cx

mov cl,dl

shl al,cl ;מזיז את הביטים שמאלה כמספר הביט בבית לדוג אם הוא מס' 4 אז אנו צריכים להזיז את הביטים 4 פעמים שמאלה כדי שהביט שאנו רוצים יהיה הכי שמאלי

pop cx ;no need to save cx

inc dl

shr al , 7 ;הביט שאנו רוצים נמצא במקום הכי שמאלי בבית ואנו רוצים שהוא יהיה במקום הכי ימני כדי שיהיה את הביט עצמו שכתוב לכן אנו מזיזים 7 פעמים שמאלה

mov INDXa, ax ;save the result in [bp+4]

pop BP DI DX CX BX AX

ret 2 ;not 4 because the bit is in there

endp

;========================================================================

;PUSH OFFSET plain after the function |

;push offset plain before the function |

;push 7 offsets (or the pc1's tables offsets or rotate's tables offsets |

;if rotate so the macro rotatecall else (if pc1) call rotateorpc1 |

;========================================================================

proc MainPc1orrotate

PushAll

push bp

mov bp,sp

mov di, [word ptr bp+32] ;offset after

mov cx,8

mov bx, [word ptr bp+28];offset 1 or pc1 or rotate

mov dl, [byte ptr di]

pc1ro0pc:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+30] ;offset before

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop pc1ro0pc

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, [word ptr bp+26];offset 2 or pc1 or rotate

mov dl, [byte ptr di]

pc1ro1pc:

push cx

mov si, [word ptr bx]

push si

mov si , [word ptr bp+30];offset before

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop pc1ro1pc

mov [byte ptr di] , dl

mov cx,8

inc di

mov bx, [word ptr bp+24];offset 3 or pc1 or rotate

mov dl, [byte ptr di]

pc1ro2pc:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+30];offset before

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop pc1ro2pc

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, [word ptr bp+22];offset 4 or pc1 or rotate

mov dl, [byte ptr di]

pc1ro3pc:

push cx

mov si, [word ptr bx]

push si

mov si , [word ptr bp+30];offset before

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop pc1ro3pc

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, [word ptr bp+20];offset 5 or pc1 or rotate

mov dl, [byte ptr di]

pc1ro4pc:

push cx

mov si, [word ptr bx]

push si

mov si , [word ptr bp+30];offset before

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop pc1ro4pc

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, [word ptr bp+18];offset 6 or pc1 or rotate

mov dl, [byte ptr di]

pc1ro5pc:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+30];offset before

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop pc1ro5pc

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, [word ptr bp+16] ;offset 7 or pc1 or rotate

mov dl, [byte ptr di]

pc1ro6pc:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+30];offset before

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop pc1ro6pc

mov [byte ptr di] , dl

pop bp

PopAll

ret 18

endp

;-------expansion----------

;====================================================

;PUSH OFFSET plain after the function |

;push offset plain before the function |

;call MainExpansion |

;mov di [bp+18] after |

;mov si,[bp+16] before because there is a push cx |

;====================================================

proc MainExpansion

PushAll

push bp

mov bp,sp

mov di, [word ptr bp+18];offset after

mov cx,8

mov bx, offset Expansion0

mov dl, [byte ptr di]

expa0:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+16];offset before

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop expa0

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, offset Expansion1

mov dl, [byte ptr di]

expa1:

push cx

mov si, [word ptr bx]

push si

mov si , [word ptr bp+16];offset before

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop expa1

mov [byte ptr di] , dl

mov cx,8

inc di

mov bx, offset Expansion2

mov dl, [byte ptr di]

expa2:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+16];offset before

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop expa2

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, offset Expansion3

mov dl, [byte ptr di]

expa3:

push cx

mov si, [word ptr bx]

push si

mov si , [word ptr bp+16];offset before

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop expa3

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, offset Expansion4

mov dl, [byte ptr di]

expa4:

push cx

mov si, [word ptr bx]

push si

mov si , [word ptr bp+16];offset before

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop expa4

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, offset Expansion5

mov dl, [byte ptr di]

expa5:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+16];offset before

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop expa5

mov [byte ptr di] , dl

pop bp

PopAll

ret

endp

;----------sboxes -------

;after sbox8 the variable with the values will be 'aftersbox'

;--------------------first s-box --------------------------

proc sbox1

push bp

mov bp,sp

mov ax, [word ptr bp+4]

xor al,al

shr ah, 2

ror ax, 1

rcl ah,4

ror ax, 1

ror al,6

mov dl, al

xor al,al

ror ah,3

ror ax, 8

cmp dl, 0

je sbox10

cmp dl, 1

je sbox11

cmp dl, 2

je sbox12

cmp dl, 3

je sbox13

sbox10:

mov bx, offset Subtation10

add bx, ax

mov al, [byte ptr bx]

jmp endpros1

sbox11:

mov bx, offset Subtation11

add bx, ax

mov al, [byte ptr bx]

jmp endpros1

sbox12:

mov bx, offset Subtation12

add bx, ax

mov al, [byte ptr bx]

jmp endpros1

sbox13:

mov bx, offset Subtation13

add bx, ax

mov al, [byte ptr bx]

endpros1:

mov bl,10

mul bl

mov [byte ptr aftersbox],al

pop bp

ret

endp

;---------------second sbox------------------------------------------------------------------------;

proc sbox2

push bp

mov bp,sp

mov ax, [word ptr bp+4]

rcl ax, 6

xor al,al

shr ah, 2

ror ax, 1

rcl ah,4

ror ax, 1

ror al,6

mov dl, al

xor al,al

ror ah,3

ror ax, 8

cmp dl, 0

je sbox20

cmp dl, 1

je sbox21

cmp dl, 2

je sbox22

cmp dl, 3

je sbox23

sbox20:

mov bx, offset Subtation20

add bx, ax

mov al, [byte ptr bx]

jmp endpros2

sbox21:

mov bx, offset Subtation21

add bx, ax

mov al, [byte ptr bx]

jmp endpros2

sbox22:

mov bx, offset Subtation22

add bx, ax

mov al, [byte ptr bx]

jmp endpros2

sbox23:

mov bx, offset Subtation23

add bx, ax

mov al, [byte ptr bx]

endpros2:

add [byte ptr aftersbox],al

pop bp

ret

endp

;-----------------------------third s-box-------------

proc sbox3

push bp

mov bp,sp

mov ax, [word ptr bp+6]

mov bx, [word ptr bp+4]

mov ah,bh

ror ax, 4

xor al,al

shr ah, 2

ror ax, 1

rcl ah,4

ror ax, 1

ror al,6

mov dl, al

xor al,al

ror ah,3

ror ax, 8

cmp dl, 0

je sbox30

cmp dl, 1

je sbox31

cmp dl, 2

je sbox32

cmp dl, 3

je sbox33

sbox30:

mov bx, offset Subtation30

add bx, ax

mov al, [byte ptr bx]

jmp endpros3

sbox31:

mov bx, offset Subtation31

add bx, ax

mov al, [byte ptr bx]

jmp endpros3

sbox32:

mov bx, offset Subtation32

add bx, ax

mov al, [byte ptr bx]

jmp endpros3

sbox33:

mov bx, offset Subtation33

add bx, ax

mov al, [byte ptr bx]

endpros3:

mov bl,10

mul bl

mov [byte ptr aftersbox+1],al

pop bp

ret

endp

;------------4 s-box-------------

proc sbox4

push bp

mov bp,sp

mov ax, [word ptr bp+4]

shl ah, 2

xor al,al

shr ah, 2

ror ax, 1

rcl ah,4

ror ax, 1

ror al,6

mov dl, al

xor al,al

ror ah,3

ror ax, 8

cmp dl, 0

je sbox40

cmp dl, 1

je sbox41

cmp dl, 2

je sbox42

cmp dl, 3

je sbox43

sbox40:

mov bx, offset Subtation40

add bx, ax

mov al, [byte ptr bx]

jmp endpros4

sbox41:

mov bx, offset Subtation41

add bx, ax

mov al, [byte ptr bx]

jmp endpros4

sbox42:

mov bx, offset Subtation42

add bx, ax

mov al, [byte ptr bx]

jmp endpros4

sbox43:

mov bx, offset Subtation43

add bx, ax

mov al, [byte ptr bx]

endpros4:

add [byte ptr aftersbox+1],al

pop bp

ret

endp

;---------------5 sbox------------------------------------------------------------------------;

proc sbox5

push bp

mov bp,sp

mov ax, [word ptr bp+4]

rcl ax, 8

xor al,al

shr ah, 2

ror ax, 1

rcl ah,4

ror ax, 1

ror al,6

mov dl, al

xor al,al

ror ah,3

ror ax, 8

cmp dl, 0

je sbox50

cmp dl, 1

je sbox51

cmp dl, 2

je sbox52

cmp dl, 3

je sbox53

sbox50:

mov bx, offset Subtation50

add bx, ax

mov al, [byte ptr bx]

jmp endpros5

sbox51:

mov bx, offset Subtation51

add bx, ax

mov al, [byte ptr bx]

jmp endpros5

sbox52:

mov bx, offset Subtation52

add bx, ax

mov al, [byte ptr bx]

jmp endpros5

sbox53:

mov bx, offset Subtation53

add bx, ax

mov al, [byte ptr bx]

endpros5:

mov bl,10

mul bl

mov [byte ptr aftersbox+2],al

pop bp

ret

endp

;-----------------------------6 s-box-------------

proc sbox6

push bp

mov bp,sp

mov ax, [word ptr bp+6]

mov bx, [word ptr bp+4]

mov ah,bh

ror ax,2

xor al,al

shr ah, 2

ror ax, 1

rcl ah,4

ror ax, 1

ror al,6

mov dl, al

xor al,al

ror ah,3

ror ax, 8

cmp dl, 0

je sbox60

cmp dl, 1

je sbox61

cmp dl, 2

je sbox62

cmp dl, 3

je sbox63

sbox60:

mov bx, offset Subtation60

add bx, ax

mov al, [byte ptr bx]

jmp endpros6

sbox61:

mov bx, offset Subtation61

add bx, ax

mov al, [byte ptr bx]

jmp endpros6

sbox62:

mov bx, offset Subtation62

add bx, ax

mov al, [byte ptr bx]

jmp endpros6

sbox63:

mov bx, offset Subtation63

add bx, ax

mov al, [byte ptr bx]

endpros6:

add [byte ptr aftersbox+2],al

pop bp

ret

endp

;-----------7 s-box---------------

proc sbox7

push bp

mov bp,sp

mov ax, [word ptr bp+4]

rcl ax, 6

xor al,al

shr ah, 2

ror ax, 1

rcl ah,4

ror ax, 1

ror al,6

mov dl, al

xor al,al

ror ah,3

ror ax, 8

cmp dl, 0

je sbox70

cmp dl, 1

je sbox71

cmp dl, 2

je sbox72

cmp dl, 3

je sbox73

sbox70:

mov bx, offset Subtation70

add bx, ax

mov al, [byte ptr bx]

jmp endpros7

sbox71:

mov bx, offset Subtation71

add bx, ax

mov al, [byte ptr bx]

jmp endpros7

sbox72:

mov bx, offset Subtation72

add bx, ax

mov al, [byte ptr bx]

jmp endpros7

sbox73:

mov bx, offset Subtation73

add bx, ax

mov al, [byte ptr bx]

endpros7:

mov bl,10

mul bl

mov [byte ptr aftersbox+3],al

pop bp

ret

endp

;---------------8 sbox------------------------------------------------------------------------;

proc sbox8

push bp

mov bp,sp

mov ax, [word ptr bp+4]

rcl ax, 10

xor al,al

shr ah, 2

ror ax, 1

rcl ah,4

ror ax, 1

ror al,6

mov dl, al

xor al,al

ror ah,3

ror ax, 8

cmp dl, 0

je sbox80

cmp dl, 1

je sbox81

cmp dl, 2

je sbox82

cmp dl, 3

je sbox83

sbox80:

mov bx, offset Subtation80

add bx, ax

mov al, [byte ptr bx]

jmp endpros8

sbox81:

mov bx, offset Subtation81

add bx, ax

mov al, [byte ptr bx]

jmp endpros8

sbox82:

mov bx, offset Subtation82

add bx, ax

mov al, [byte ptr bx]

jmp endpros8

sbox83:

mov bx, offset Subtation83

add bx, ax

mov al, [byte ptr bx]

endpros8:

add [byte ptr aftersbox+3],al

pop bp

ret

endp

;------------------ip-----------

;PUSH OFFSET plain after the function

;push offset plain before the function

;all offsets of ip's tables or ip-1's tables

;call MainIp

;mov di [bp+34] after function

;mov bx,[bp+32] before function

proc MainIp

PushAll

push bp

mov bp,sp

mov di, [word ptr bp+34]

mov cx,8

mov bx, [word ptr bp+30]

mov dl, [byte ptr di]

laip0:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+32]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop laip0

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, [word ptr bp+28]

mov dl, [byte ptr di]

laip1:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+32]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop laip1

mov [byte ptr di] , dl

mov cx,8

inc di

mov bx, [word ptr bp+26]

mov dl, [byte ptr di]

laip2:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+32]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop laip2

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, [word ptr bp+24]

mov dl, [byte ptr di]

laip3:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+32]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop laip3

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, [word ptr bp+22]

mov dl, [byte ptr di]

laip4:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+32]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop laip4

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, [word ptr bp+20]

mov dl, [byte ptr di]

laip5:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+32]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop laip5

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, [word ptr bp+18]

mov dl, [byte ptr di]

laip6:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+32]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop laip6

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, [word ptr bp+16]

mov dl, [byte ptr di]

laip7:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+32]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop laip7

mov [byte ptr di] , dl

pop bp

PopAll

ret 20

endp

;push after function

;push before function

;call MainPc2

proc MainPc2

PushAll

push bp

mov bp,sp

mov di, [word ptr bp+18]

mov cx,8

mov bx, offset PC20

mov dl, [byte ptr di]

lapc20:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+16]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop lapc20

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, offset PC21

mov dl, [byte ptr di]

lapc21:

push cx

mov si, [word ptr bx]

push si

mov si , [word ptr bp+16]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop lapc21

mov [byte ptr di] , dl

mov cx,8

inc di

mov bx, offset PC22

mov dl, [byte ptr di]

lapc22:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+16]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop lapc22

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, offset PC23

mov dl, [byte ptr di]

lapc23:

push cx

mov si, [word ptr bx]

push si

mov si , [word ptr bp+16]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop lapc23

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, offset PC24

mov dl, [byte ptr di]

lapc24:

push cx

mov si, [word ptr bx]

push si

mov si , [word ptr bp+16]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop lapc24

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, offset PC25

mov dl, [byte ptr di]

lapc25:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+16]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop lapc25

mov [byte ptr di] , dl

pop bp

PopAll

ret

endp

;push after function

;push before function

;call MainPer

proc MainPer

PushAll

push bp

mov bp,sp

mov di, [word ptr bp+18]

mov cx,8

mov bx, offset Permutation0

mov dl, [byte ptr di]

lapermutation20:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+16]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop lapermutation20

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, offset Permutation1

mov dl, [byte ptr di]

lapermutation21:

push cx

mov si, [word ptr bx]

push si

mov si , [word ptr bp+16]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop lapermutation21

mov [byte ptr di] , dl

mov cx,8

inc di

mov bx, offset Permutation2

mov dl, [byte ptr di]

lapermutation22:

push cx

mov si, [word ptr bx]

push si

mov si ,[word ptr bp+16]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop lapermutation22

mov [byte ptr di] , dl

inc di

mov cx,8

mov bx, offset Permutation3

mov dl, [byte ptr di]

lapermutation23:

push cx

mov si, [word ptr bx]

push si

mov si , [word ptr bp+16]

push si

inc bx

call getBit

pop ax

shl dl,1

add dl, al

pop cx

loop lapermutation23

mov [byte ptr di] , dl

pop bp

PopAll

ret

endp

;================================================

;all Encryption function Include the all 16 rounds

;to call this proc you need to do this:

;mov sp,100h mov the stack pointer to the start

;call ENCRYPTION1

;before the end of the proc:

; mov sp,254 to put the stack pointer on the ip of the rest of the main action

;ret

;endp

;the result of the encryption will be in the variable Encryptionresult and the proc will print the result

;on the window the print text will be showed at ascii codes

;================================================

proc ENCRYPTION1

Encryption:

push offset textafterip

push offset hextext

push offset ip0

push offset ip1

push offset ip2

push offset ip3

push offset ip4

push offset ip5

push offset ip6

push offset ip7

call MainIp

push offset KeyAfterPC1Beforepc2beforerotateandmain

push offset hexkey

push offset PC10

push offset PC11

push offset PC12

push offset PC13

push offset PC14

push offset PC15

push offset PC16

call MainPc1orrotate

mov ax, [word ptr textafterip]

mov [word ptr Leftsidetext],ax

mov ax, [word ptr textafterip+2]

mov [word ptr Leftsidetext+2],ax

mov ax, [word ptr textafterip+4]

mov [word ptr Rightsidetext],ax

mov ax, [word ptr textafterip+6]

mov [word ptr Rightsidetext+2],ax

mov si,0

MainEncryption:

Ffunction1

jmp labffunc2

betweenlab:

jmp MainEncryption

labffunc2:

Ffunction2

inc si

cmp si,16

jae labcontinue

jmp MainEncryption

labcontinue:

mov bx, offset textafterip

mov ax, [word ptr Leftsidetext]

mov [word ptr bx+4],ax

mov ax, [word ptr Leftsidetext+2]

mov [word ptr bx+6],ax

mov ax, [word ptr Rightsidetext]

mov [word ptr bx],ax

mov ax, [word ptr Rightsidetext+2]

mov [word ptr bx+2],ax

push offset Encryptionresult

push offset textafterip

push offset ip\_10

push offset ip\_11

push offset ip\_12

push offset ip\_13

push offset ip\_14

push offset ip\_15

push offset ip\_16

push offset ip\_17

call MainIp

mov ah,9

mov dx,offset encrypres

int 21h

mov dl,[byte ptr Encryptionresult]

mov ah,2

int 21h

mov dl,[byte ptr Encryptionresult+1]

mov ah,2

int 21h

mov dl,[byte ptr Encryptionresult+2]

mov ah,2

int 21h

mov dl,[byte ptr Encryptionresult+3]

mov ah,2

int 21h

mov dl,[byte ptr Encryptionresult+4]

mov ah,2

int 21h

mov dl,[byte ptr Encryptionresult+5]

mov ah,2

int 21h

mov dl,[byte ptr Encryptionresult+6]

mov ah,2

int 21h

mov dl,[byte ptr Encryptionresult+7]

mov ah,2

int 21h

mov sp,254

ret

endp

;================================================

;all Decryption function Include the all 16 rounds

;to call this proc you need to do this:

;mov sp,100h mov the stack pointer to the start

;call ENCRYPTION1

;before the end of the proc:

; mov sp,254 to put the stack pointer on the ip of the rest of the main action

;ret

;endp

;the result of the decryption will be in the variable Decryptionresult and the proc will print the result

;on the window the print text will be showed at ascii codes

;================================================

proc Decryption1

Decrypted:

ChangeRandL

push offset textafterip

push offset hextext

push offset ip0

push offset ip1

push offset ip2

push offset ip3

push offset ip4

push offset ip5

push offset ip6

push offset ip7

call MainIp

push offset KeyAfterPC1Beforepc2beforerotateandmain

push offset hexkey

push offset PC10

push offset PC11

push offset PC12

push offset PC13

push offset PC14

push offset PC15

push offset PC16

call MainPc1orrotate

mov ax, [word ptr textafterip]

mov [word ptr Leftsidetext],ax

mov ax, [word ptr textafterip+2]

mov [word ptr Leftsidetext+2],ax

mov ax, [word ptr textafterip+4]

mov [word ptr Rightsidetext],ax

mov ax, [word ptr textafterip+6]

mov [word ptr Rightsidetext+2],ax

PushAll

mov ax,0

mov bx,offset KeyDecTemp

mov si,offset KeyAfterPC1Beforepc2beforerotateandmain

SaveKeyafterpc1:

mov cl,[byte ptr si]

inc si

mov [byte ptr bx], cl

inc bx

inc ax

cmp ax, 7

je labhere1Decsave

jmp SaveKeyafterpc1

labhere1Decsave:

PopAll

mov si,0

MainDec:

Ffunction1D

jmp labffunc2Dec

betweenlabDec:

jmp MainDec

labffunc2Dec:

Ffunction2D

inc si

cmp si, 16

je labcontinue1Dec

jmp MainDec

labcontinue1Dec:

mov bx, offset textafterip

mov ax, [word ptr Leftsidetext]

mov [word ptr bx+4],ax

mov ax, [word ptr Leftsidetext+2]

mov [word ptr bx+6],ax

mov ax, [word ptr Rightsidetext]

mov [word ptr bx],ax

mov ax, [word ptr Rightsidetext+2]

mov [word ptr bx+2],ax

push offset Decryptionresult

push offset textafterip

push offset ip\_10

push offset ip\_11

push offset ip\_12

push offset ip\_13

push offset ip\_14

push offset ip\_15

push offset ip\_16

push offset ip\_17

call MainIp

mov ah,9

mov dx,offset decrypres

int 21h

mov dl,[byte ptr Decryptionresult]

mov ah,2

int 21h

mov dl,[byte ptr Decryptionresult+1]

mov ah,2

int 21h

mov dl,[byte ptr Decryptionresult+2]

mov ah,2

int 21h

mov dl,[byte ptr Decryptionresult+3]

mov ah,2

int 21h

mov dl,[byte ptr Decryptionresult+4]

mov ah,2

int 21h

mov dl,[byte ptr Decryptionresult+5]

mov ah,2

int 21h

mov dl,[byte ptr Decryptionresult+6]

mov ah,2

int 21h

mov dl,[byte ptr Decryptionresult+7]

mov ah,2

int 21h

mov sp,254

ret

endp

;==============================================

;the start choises like how to write the text and the key and the nput text and input key

;no parameters

;===============================================

proc keyandtextstartscreen

morechance:

mov dx, offset mes1

mov ah,9

int 21h

mov ah,1

int 21h

call NewLine

cmp al, '1'

je labencrypcont

cmp al, '2'

je labencrypcont

cmp al,'3'

je Readfromfiletotext

mov dx, offset mes3

mov ah,9

int 21h

jmp morechance

Readfromfiletotext:

call Readfromfile

jmp Keycheckifletterorhex

labencrypcont:

cmp al,'1'

jne textinletter

mov dx, offset mes4

mov ah,9

int 21h

mov dx,offset plaintextkelet

mov bx,dx

mov [byte ptr bx], 9

mov ah,0Ah

int 21h

call NewLine

call RemoveEntertext

hextextkelet

jmp Keycheckifletterorhex

textinletter:

mov dx, offset mes4

mov ah,9

int 21h

mov dx,offset plaintextkelet

mov bx,dx

mov [byte ptr bx], 9

mov ah,0Ah

int 21h

call NewLine

call RemoveEntertext

Keycheckifletterorhex:

mov dx, offset mes2

mov ah,9

int 21h

mov ah,1

int 21h

call NewLine

cmp al, '1'

je labencrypconk

cmp al, '2'

je labencrypconk

mov dx, offset mes8

mov ah,9

int 21h

jmp Keycheckifletterorhex

labencrypconk:

cmp al,'1'

jne keyinletter

mov dx, offset mes5

mov ah,9

int 21h

mov dx,offset plainkeykelet

mov bx,dx

mov [byte ptr bx], 9

mov ah,0Ah

int 21h

call NewLine

call RemoveEnterkey

hexkeykelet

jmp endprocofscreen

keyinletter:

mov dx, offset mes4

mov ah,9

int 21h

mov dx,offset plainkeykelet

mov bx,dx

mov [byte ptr bx], 9

mov ah,0Ah

int 21h

call NewLine

call RemoveEnterkey

endprocofscreen:

ret

endp

start:

mov ax, @data

mov ds, ax

; --------------------------

; Your code here

;when you push for the procs the offset after is before offset before'

mov ax,3h

int 10h

;start of print the start window

mov ah,9

mov dx,offset des02

int 21h

mov ah,9

mov dx,offset des03

int 21h

mov ah,9

mov dx,offset des04

int 21h

mov ah,9

mov dx,offset des05

int 21h

mov ah,9

mov dx,offset des06

int 21h

mov ah,9

mov dx,offset des07

int 21h

mov ah,9

mov dx,offset des08

int 21h

mov ah,9

mov dx,offset des09

int 21h

mov ah,9

mov dx,offset des10

int 21h

mov ah,9

mov dx,offset des11

int 21h

mov ah,9

mov dx,offset des12

int 21h

mov ah,9

mov dx,offset des13

int 21h

mov ah,9

mov dx,offset des14

int 21h

mov ah,9

mov dx,offset des15

int 21h

mov ah,9

mov dx,offset des16

int 21h

mov ah,9

mov dx,offset des17

int 21h

mov ah,9

mov dx,offset des18

int 21h

mov ah,9

mov dx,offset des19

int 21h

mov ah,9

mov dx,offset des21

int 21h

;end of print the start window

call openfile

no1or2:

mov dx, offset mes0

mov ah,9

int 21h

mov ah,1

int 21h

call NewLine

cmp al,'1'

je Encryp

cmp al,'2'

je Decryp

mov dx, offset mes8

mov ah,9

int 21h

jmp no1or2

;;;;;;;;;;;;;;;;;;;;

Encryp: ;encryption

call keyandtextstartscreen

mov sp, 100h

call ENCRYPTION1

PushAll

mov ax,0

mov bx,offset encriptionordecryptionresult

mov si,offset Encryptionresult

movencypresulttothefilevar:

mov cl,[byte ptr si]

inc si

mov [byte ptr bx], cl

inc bx

inc ax

cmp ax, 8

je labcontinueaftermov

jmp movencypresulttothefilevar

labcontinueaftermov:

PopAll

call DeleteFile ;delete the previous file

call CreateFile ;create new file

call WriteToFile ;write the encryption result that in the variable encriptionordecryptionresult 64 bits in the file

jmp exit

Decryp:

call keyandtextstartscreen

mov sp, 100h

call Decryption1

PushAll

mov ax,0

mov bx,offset encriptionordecryptionresult

mov si,offset Decryptionresult

movdecrypresulttothefilevar:

mov cl,[byte ptr si]

inc si

mov [byte ptr bx], cl

inc bx

inc ax

cmp ax, 8

je labcontinueaftermovatdecryp

jmp movdecrypresulttothefilevar

labcontinueaftermovatdecryp:

PopAll

call DeleteFile

call CreateFile

call WriteToFile

; --------------------------

exit:

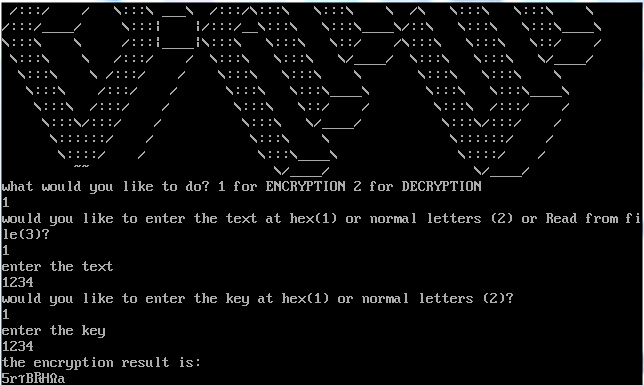
mov ax, 4c00h

int 21h

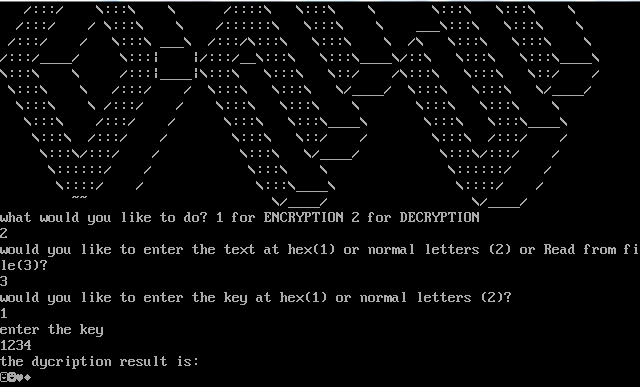
END start

# דוג' הרצה

**הצפנה של טקסט: 1234H ומפתח 1234H**

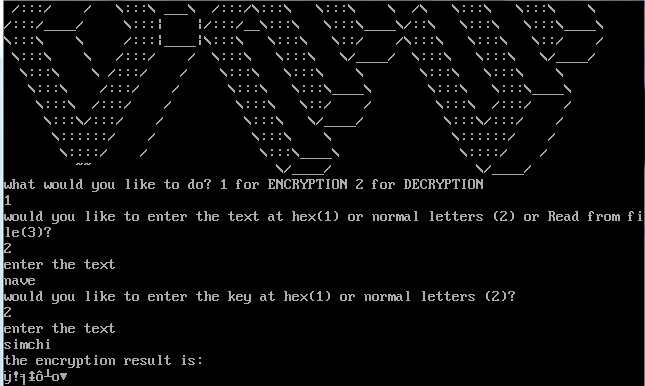


**פענוח של התוצאה למעלה ומפתח 1234H**

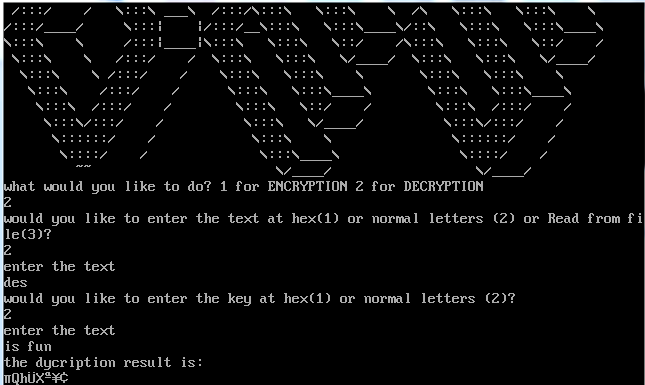


**ניתן לראות שהתוצאה של הפענוח היא קודי ההסקי של 1234**

**הצפנה של טקסט: nave ומפתח: simchi**



**פענוח של טקסט: des ומפתח: is fun**



# סיכום אישי

אני נהניתי מאוד לעשות את הפרוייקט, נהניתי ללמוד את ההצפנה, היה לי כיף לבצע אותה, אך קשה, היו אתגרים שהייתי צריך לפתור בעצמי, כמו לדעת לעבוד עם ביט בודד, זה לא היה קל, אך במבט לאחור אני מבין שזה היה שווה את זה, אני יכול להגיד כי מה שאני עשיתי לא הרבה ילדים עשו ואני גאה בעצמי על זה.

לסיכום הפרוייקט היה חלק משמעותי מחיי, סדר יומי השתנה, אך אני שמח שבחרתי בנושא זה ולא ויתרתי.