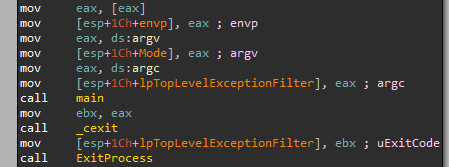
Reverse HW 2

**Goblin**

תחילה, כאשר ניגשנו לקובץ ניסינו להבין איפה מתחיל הmain. ניסינו להסתכל על המחרוזות בתכנית, אך לא הצלחנו באופן מובהק להבין מהן איזו פונקציה היא הmain, ולכן בסופו של דבר הסקנו איזה פונקציה היא הmain לפי קריאה של הstart. חיפשנו את הפונקציה שמקבלת את envp, argv, argc, ובעזרת מידע נוסף מהIDA הגענו לפונקציה הנכונה:



לאחר מכן, הסתכלנו על תחילת הריצה של main. תחילה, היא קוראת לפונקציה שלא ממש משפיעה על הלוגיקה של התכנית. לאחר מכן, main קוראת לפונקציה שבהמשך נתנו לה את השם PrepareMaze.

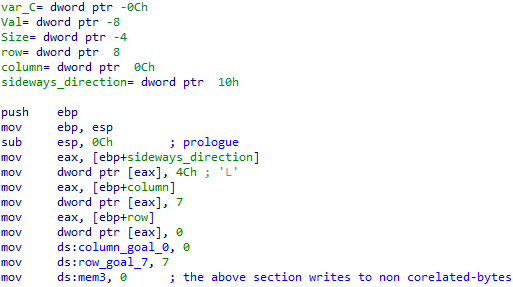
הפונקציה PrepareMaze: ניתן לראות כי הפונקציה כותבת לרצף של 64 בתים בזכרון.  
אחרי בחינה מעמיקה, הגענו למסקנה שמה שהיא כותבת לזכרון נראה ככה:

A picture containing blue, electric blue, pattern, majorelle blue

Description automatically generated

תחילה חשבנו שהרצף הזה מייצג את המפתח שאנחנו מחפשים, ושבמהלך ריצת התכנית מתמלאים "הערכים הנכונים" . לקראת סוף הניתוח של הקובץ, הבנו שמדובר במבוך.

בנוסף, על ידי התבוננות בחלק הבא של הפונקציה:



הגענו למסקנה שהפונקציה מקבלת כפרמטרים שני מצביעים ל-int ומצביע ל-char. בהמשך הבנו את מטרת הפרמטרים הללו – בעזרתם הפונקציה מאתחלת את המיקום ההתחלתי שלנו במבוך (שורה ועמודה) ואת הכיוון ההתחלתי שבו אנו זזים (בתמונה זהו התו ‘L’).

הפונקציה מאתחלת (בלי קשר לפרמטרים) את מיקום המטרה במבוך, ואת מונה הדגלים.

בכך סיימנו את ניתוח הפונקציה PrepareMaze.

המשך main:

בתחילת main, מאותחלים שני משתנים. הסקנו שתפקידיהם הם של counter ו-limit של לולאה, וכעת הגענו לניתוח תוכן הלולאה.

הלולאה רצה עד 7 פעמים. איטרציה של הלולאה נראית כך:

תחילה, קוראים לפונקציה שמאוחר יותר קראנו לה GetCommand. נפרט על ניתוחה כעת:

GetCommand:

הפונקציה מקבלת שני פרמטרים – char\*, int\*.

ה-char\* משמש לאחסון ה-command שניתן על ידי המשתמש. ה-int\* משמש לאחסון מספר.

הפונקציה קולטת תו מהמשתמש ושומרת אותו בפרמטר ה-char\*. אחרי כן, יש קריאה לפונקציה נוספת. בהתחלה לא כל כך הבנו מה הפונקציה הנוספת עושה, אך כן הבנו שהיא לא משפיעה על הלוגיקה של התכנית(מאוחר יותר גילינו שהיא מקודדת את המפתח בהתאם למהלכים במבוך).

לאחר קליטת תו מהמשתמש ב-GetCommand – ראינו כי אם התו אינו ‘C’ או ‘D’ או ‘U’ – ריצת התכנית מסתיימת.  
אם התו הוא ‘C’, אז קוראים תו נומרי שמתורגם למספר בין 0(כולל) ל-9(כולל).

המשך הלולאה ב-main: יש שני מסלולים בלולאה, וההחלטה לאיזה מסלול לגשת נקבעת על פי התו שנקלט ב-GetCommand.

מסלול א' – אם התו שנקלט אינו ‘C’:

נקראת הפונקציה שמאוחר יותר קראנו לה MoveRow. נפרט עליה כעת.

הפונקציה MoveRow: מקבלת int\*, int, char.

בעזרת הפרמטרים המספריים, הפונקציה מחשבת את המיקום הנוכחי במבוך ובודקת שאין במיקום זה את התו ‘X’ או ‘.’(נקודה). אחרת התכנית מסתיימת.  
אחר כך, בהתאם להאם התו הוא ‘U’ או ‘D’, משנים את ערך הארגומנט הראשון לפי התו שנמצא במיקום הנוכחי.  
בשלב זה של הניתוח, עדיין חשבנו שהמבוך הוא מפתח, אך התחלנו לפקפק בכך, כי הייתה לנו הרגשה ש-U ו-D מציינים Up או Down בהתאמה, לפי הצורה שבה משנים את הארגומנט הראשון (מספר שורה).

לאחר שינוי הארגומנט הראשון, קוראים איתו לפונקציה אחרת, שקראנו לה PathBlockCheck. פונקציה זו בודקת שהמיקום החדש הוא בתחומי הלוח ושהמיקום החדש אינו מיקום של תו ‘X’. אחרת התכנית מסתיימת.

עברנו כעת למסלול האחר בלולאה ב-main. מסלול זה מתרחש אם התו שנקלט מהמשתמש הוא ‘C’. במסלול זה נקראת פונקציה חדשה, שקראנו לה מאוחר יותר MoveInSidewayDirection.  
הפונקציה MoveInSidewayDirection: מקבלת ארבעה פרמטרים – int\*, int\*, char\*, int.

מאוחר יותר הבנו שהפרמטרים מהווים את מספר השורה, מספר העמודה, כיוון התזוזה, ומספר הצעדים.

הפונקציה קוראת לעצמה באופן רקורסיבי כמספר הצעדים. בכל צעד רקורסיבי, היא בודקת את כיוון התנועה ולפיו היא משנה את מספר העמודה. אם הגענו לקצה המבוך, אז משנים את מספר השורה ואת הכיוון. בנוסף, אם עוברים במשבצת שבה התו ‘F’, מקדמים את מונה הדגלים.

בשלב זה התמונה התבהרה לנו, והבנו כי מדובר במבוך ולא במפתח, וכי R\L\D\U הם כיווני תנועה. בנוסף, הבנו כי C והמספר מסמלים צעדים \ התקדמות.

מכאן, חזרנו אחורה, התחלנו לתת שמות למשתנים\לפונקציות, ולפענח את חוקי המשחק. בנוסף, סיימנו לפענח את הלולאה ב-main, שבודקת בסופה האם הגענו למשבצת הסיום, והאם אספנו שני דגלים.

חוקי המשחק שהבנו הם:

* אנחנו נמצאים בתוך מבוך 8x8, מתחילים במשבצת [0,7], וכיוון התנועה הוא שמאלה.
* יש במשחק עד שבעה מהלכים.
* בכל שלב, נותנים אחת משלוש פקודות :

1. U
2. D
3. C + Number

אם הפקודה היא U או D : מסתכלים על התו במיקום הנוכחי במבוך. עולים או יורדים מספר שורות כמספר התו.

אם הפקודה היא C + Number, אז מתקדמים לפי כיוון התנועה הנוכחי, כ-Number צעדים. במידה ומתקדמים מעבר לגבולות הלוח, אז משנים את כיוון התנועה, ועולים שורה במקום להתקדם לצדדים. לאחר עליית השורה ממשיכים להתקדם את יתרת הצעדים. תוך כדי ההתקדמות, בודקים האם עברנו במשבצת שיש בה דגל 'F', ומקדמים את מונה הדגלים בהתאם.

* כאשר מגיעים למשבצת [7,0], בודקים האם אספנו שני דגלים ואם כן, מודפס מפתח ההצפנה הסופר סודי. 😊

בהתחלה, חשבנו שצריך לעצור בכל משבצת שיש בה F אם אנחנו רוצים לאסוף את הדגל, ומספר המהלכים לא הספיק לנו תחת המגבלות הללו. לכן שינינו את קובץ הריצה כך שיהיו לנו יותר מהלכים.

בהבסת המשחק (לאחר ששינינו את הקובץ) התקבל לנו המפתח הבא:



המפתח שקיבלנו היה נראה לנו משונה, ולכן חזרנו אחורה והבחנו שאפשר לאסוף דגלים במהלך תנועה. פתרנו את המבוך בשבעה צעדים, וקיבלנו את המפתח הבא:

A black background with white text

Description automatically generated with low confidence

ההבדל בין המפתחות השונים שקיבלנו נובע מכך שהמפתח נוצר על בסיס הפקודות שאנחנו מכניסים.