פרויקט גמר באסמבלר

מגיש: דויד קיסר שמידט

תעודת זהות: 325594224

כיתה: יב' 5

בית הספר: תיכון ליד האוניברסיטה

מורה מנחה**:** אורנה דניאלי

Diagram

Description automatically generatedשנת לימוד: 2021-2022

תוכן עניינים

[מבוא 3](#_Toc104656096)

[הוראות הפעלה וסיסמאות 4](#_Toc104656097)

[משתנים בתכנית – Macros 5](#_Toc104656098)

[משתנים בתכנית – Globals 5](#_Toc104656099)

[טבלת פרוצדורות 8](#_Toc104656100)

[תרשים זרימה כללי של הפרויקט 11](#_Toc104656101)

[תרשימי זרימה של פרוצדורות 12](#_Toc104656102)

[אלגוריתמים מיוחדים 24](#_Toc104656103)

[הצגת תמונה על המסך 24](#_Toc104656104)

[הפיכת תמונה לתמונת ASCII 24](#_Toc104656105)

[המרה של תמונה לפורמט bmp 25](#_Toc104656106)

[סיכום ורפלקציה 27](#_Toc104656107)

[קוד הפרויקט 28](#_Toc104656108)

[קובץ הקבועים menu.asm 28](#_Toc104656109)

[הקובץ הראשי base.asm 29](#_Toc104656110)

# מבוא

אסמבלי היא שפה מאוד נמוכה, שמצד אחד מאפשרת שליטה מאוד טובה ברכיבי החומרה ובתקשורת עם מערכת ההפעלה, אך מצד שני, אינה מאפשרת תקשורת בשפה אנושית עם המחשב, כמו שפות גבוהות כדוגמת פייתון. עיבוד תמונה בשפות גבוהות ניתן לבצע באופן מאוד אלגנטי, באמצעות הספרייה numpy, ומשימות שנראות מסובכות, ניתן לבצע בכמה שורות בודדות. באסמבלי לעומת זאת, יש לממש הכל מאפס, ולכן אפילו משימות פשוטות, דורשות הרבה מאמץ. אחד הדברים שהשפות הגבוהות ״מסתירות״ מאיתנו, הוא הדרך שבה הן קוראות את התמונות, לכן החלטתי שאעסוק בעיבוד תמונה, כדי להבין כיצד הדבר מתבצע.

בפרוייקט זה החלטתי שאני אייצג תמונה כטקסט, ואציג את התמונה על המסך. כדי לבצע זאת, צריך להבין כיצד מיוצגת תמונה במחשב. המחשב מייצג תמונה כמטריצה שכל תא בה הוא שלשה של ערכים RGB שכל אחד מהם הוא בין 0 ל-255. שמהם ניתן להרכיב הרבה מאוד צבעים. אני החלטתי להתמקד בתמונות שמכילות ערכי אפור, כיוון שמחרוזת מכילה אך ורק שחור לבן. תמונות כאלה, מכילות שלשת ערכים זהה, עם ערך בין 0 ל-255, כאשר ערך גבוה מציין צבע בהיר יותר, וערך נמוך מציין צבע כהה יותר. את מלאכת הצגת התמונה על המסך, ביצעתי על ידי המרה של התמונות מראש לפורמט BMP, שזה פורמט יחסית נוח לשימוש, בשונה מ- JPEG ו-PNG אשר מורכבים יותר ודורשים דיוק נומרי, אותו קשה להשיג באסמבלי. כדי להציג את התמונה בפורמט הנל, נעזרתי [בספר של מרכז הסייבר](https://data.cyber.org.il/assembly/assembly_book.pdf) שהכיל הסברים רבים שתרמו לי רבות בעת כתיבת הקוד, על כן אני רוצה להודות לכותבי הספר. כמו כן, בעת הגדרת הטרנספורמציה מערכי האפור להאסקי, השתמשתי בתבנית בסיסית, אותה נמצא למצוא [כאן](https://www.jonathan-petitcolas.com/2017/12/28/converting-image-to-ascii-art.html).

בחרתי בנושא זה, כיוון שהוא נוגע בעיבוד תמונה, נושא מאוד קרוב אלי, אותו למדתי בקפידה השנה. כמו כן, רציתי להציג תמונה באופן שיהיה נוח לשמור אותה, פשוט כרצף תווים, ולכן הצגה כתווי ascii היה עבורי פתרון נהדר, שכן השמירה מתבצעת בקובץ טקסט פשוט, מה שדורש פחות מקום, וקל יותר לפתיחה. בנוסף, הצגה של תמונה כמחרוזת הרבה פעמים יכולה לתעתע את המשתמש, שכן, אם מתקבלת המחרוזת בלבד, מבלי שידוע שהיא מייצגת תמונה, מבלי לעשות zoom out קשה להבין למה התכוון המשורר, ולכן אפשר ממש ליצור חידות שלמות מרצף תווים זה, כמו למשל, אם משמיטים את ירידות השורה בקובץ הטקסט, המשתמש יכול לעבור על כל האפשרויות של אורכי השורות עד שמתקבלת תמונה הגיונית, וכך, נוצרת חידה שלמה! יתרה מכך, ניתן לקחת צילום מסך של טקסט, להמיר אותו לתמונת האסקי, ואז להעביר למשתמש בלי ירידות שורה, ואז מדובר ממש בהצפנה, שאמנם ניתן לפרוץ די בקלות, אך זו עדיין חידה נחמדה.

לבסוף, ברצוני להודות למורת המגמה למדעי המחשב, אורנה דניאלי.

# הוראות הפעלה וסיסמאות

**קובץ מפעיל**: base.asm מכיל את כל הקוד של התכנית.

**קובץ נלווה**: menu.asm מכיל קבועים, מייבא אותו הקובץ הראשי.

**הרצת התכנית**: על מנת להריץ את התכנית יש להריץ את הפקודות הבאות ב-dosbox:

1. mount c: c:\
2. c:
3. cd tasm\bin
4. cycles=max
5. tasm /zi base.asm
6. tlink /v base.obj
7. base

התכנית מציגה מסך פתיחה למשתמש, שם הוא יכול לבחור אם לעבור לתפריט הראשי, או לסגור את התכנית. כדי לעבור לתפריט, עליו להקיש על המקש m או ללחוץ עם העכבר על הכפתור השחור בפינה השמאלית התחתונה שעל המסך.

לאחר מכן, אם נבחר התפריט הראשי, הוא מוצג, ועל המשתמש להקיש נתיב לתמונת bmp. תמונות bmp שהכנתי מראש, שכן מלאכת ההכנה שלהן, קצת מייגעת, הן:

1. ascii.bmp
2. asciiLog.bmp
3. Error.bmp
4. log.bmp
5. p.bmp

A picture containing table

Description automatically generatedעל התמונה להיות בגדלים 200 שורות על 320 עמודות, ובצבעי grayscale. הסיבה למימדים אלה היא שהגודל המקסימלי של המסך הוא בדיוק המימדים הנ״ל. השימוש ב-grayscale הוא בגלל שהמרה מתמונה צבעונית לתמונה אפורה, דורשת כפל במטריצה הבאה:

היות שבשפת TASM אין רג'יסטרים יעודים ל-floating point החלטתי לקבל תמונות בערכי אפור בלבד.

לאחר הכנסת הנתיב, התכנית תבדוק אם הנתיב חוקי, כלומר, האם הקובץ קיים, והאם הסיומת היא bmp, אם נמצא שהוא לא חוקי, היא תציג מסך שגיאה למשתמש, ותבקש ממנו לבחור, על ידי לחיצה על מקש במקלדת, האם לעצור את התכנית (המקש (q, או לחזור לתפריט הראשי להקשת נתיב פעם נוספת (המקש (c. לבסוף, כאשר התקבל נתיב חוקי, התכנית תפתח את הקובץ, ותפענח אותו לפי פורמט ה-bmp. עם סיום הפענוח, היא תציג את התמונה שהתקבלה על המסך, כאן חשוב לציין, שבמידה שהתמונה מכילה ערכי צבע, שאינם אפור, התמונה שתתקבל לא תהיה ייצוגית. לאחר ההצגה, היא תיצור קובץ חדש על מנת לשמור את ההמרה להאסקי. אם הפתיחה נכשלה, התכנית תעצור. אחרת, היא תעבור על התמונה פעם נוספת, והפעם, תמיר כל ערך אפור לערך האסקי, על פי הטרנספורמציה שנבנתה מראש. כל זה קורה כאשר התמונה עוד מוצגת על המסך, על מנת לחזור לתפריט הראשי, יש להקיש על תו כלשהו במקלדת. לאחר הקש התו, היא תשמור את המידע בקובץ חדש, ששמו יהיה מספר הקובץ שהתכנית יצרה, ואחריו asciiGe.txt. כדי לראות את תוכן הקובץ ולהיווכח שאכן הוא מייצג את התמונה, כדאי לעשות zoom out.

ניתן לחזור על התהליך 10 פעמים, אחריו התכנית תעצור.

לאחר כל יצירת קובץ, בסוף הבקשה הנוכחית, התכנית תסגור את הקובץ. למעשה, בכל הצגה של קובץ, הקובץ נסגר, ונפתח מחדש לאחר מכן.

## משתנים בתכנית – Macros

1. PRESSED - קיצור לערך שמציין האם מקש נלחץ.
2. NOT\_PRESSED – קיצור לערך מסמן שמקש לא נלחץ.
3. MENU\_CHAR – קיצור למקש שדרכו אפשר לעבור לתפריט הראשי.
4. QUIT\_CHAR – קיצור למקש שממנו אפשר לסגור את התכנית.
5. QUIT\_OPTION – ערך החזרה של פרוצדורה שמסמן שצריך לסגור את התכנית.
6. MENU\_COLOR – ערך שמסמן את צבע התפריט.
7. START\_MENU\_ROW – השורה בה מתחיל התפריט הראשי.
8. START\_MENU\_COL – העמודה בה מתחיל התפריט הראשי.
9. INVALID\_PATH\_VAL – ערך החזרה שמסמן שהנתיב שהמשתמש הכניס, אינו חוקי.
10. CONT\_CHAR – התו שמסמן שהמשתמש רוצה להמשיך, למרות שהקיש נתיב לא חוקי.
11. VALID\_FILE – ערך החזרה שמסמל שהקובץ חוקי.
12. BMP\_END\_SIZE – גודל הסיומת של נתיב לקובץ bmp.
13. BMP\_COLS\_AMOUNT – כמות העמודות התמונת bmp.
14. BMP\_ROWS\_AMOUNT – כמות השורות בתמונת bmp.
15. BMP\_HEADER\_SIZE – גודל ה-header בתמונת bmp.
16. BMP\_PALETTE\_SIZE – גודל פלטת הצבעים בתמונת bmp.
17. SCREEN\_START\_OFFSET – המקום במסך אליו צריך לכתוב את המידע של התמונה.
18. PATH\_LENGTH – גודל הנתיב המקסימלי שאפשר להקיש.
19. OUTPUT\_PATH\_LENGTH – גודל נתיב הפלט.
20. MAX\_FILES - מספר הפעמים המקסימלי שאפשר ליצור תמונות ascii בהרצה אחת של התכנית.
21. arg1 – קיצור למשתנה הראשון שמוכנס לפרוצדורה.
22. arg2 - קיצור למשתנה השני שמוכנס לפרוצדורה.
23. arg3 - קיצור למשתנה השלישי שמוכנס לפרוצדורה.
24. arg4 - קיצור למשתנה הרביעי שמוכנס לפרוצדורה.
25. arg5 - קיצור למשתנה החמישי שמוכנס לפרוצדורה.

## משתנים בתכנית – Globals

1. NUMBER\_OF\_FILES – מספר הקבצים שהתכנית יצרה עד כה במהלך ריצתה.
2. PaletteBuffer – מערך של תווים שמטרתו לשמור את פלטת הצבעים של התמונה.
3. \_image\_name – נתיב לתמונת הפתיח שמוצגת על המסך בעת הרצת התכנית.
4. \_error\_image – נתיב לתמונת השגיאה, שמוצגת כאשר המשתמש מבצע הוראה לא תקינה.
5. \_open\_menu – מחרוזת הטקסט שמוצג בתפריט הראשי.
6. Menu – מחרוזת גרפית שמטרתה להציג את הלוגו של האפליקציה. מוצגת בתפריט הראשי ושמורה בקובץ menu.asm.
7. Ascii\_table – טבלת ההמרה מערכי אפור לערכי ascii. גודלה 256, וכדי להמיר ערך אפור i פשוט ניגשים לטבלא במקום ה-i, ולוקחים את הערך שנמצא שם.
8. output\_file\_name – שם קובץ הפלט.
9. Len – מימדי הריבוע השחור בתחתית המסך.
10. y\_coordinate – השורה ממנה מתחיל הריבוע.
11. y\_temp – משתנה שמכיל את השורה נוכחית כשצובעים בריבוע.
12. x\_coordinate – העמודה ממנה מתחיל הריבוע.
13. x\_temp – משתנה שמכיל את העמודה הנוכחית כשצובעים את הריבוע.
14. Color – צבע הריבוע בפינה השמאלית התחתונה בתפריט הפתיחה.
15. CURRENT\_LINE\_LENGTH – אורך השורה בקובץ הפלט, אותה צריך לכתוב לקובץ, זה 321 ולא 320 כדי להוסיף ירידת שורה.
16. input\_file\_handle – משתנה ששומר את אובייקט ה-handle שהתקבל ממערכת ההפעלה לאחר פתיחת קובץ הקלט.
17. input\_file\_path – מערך ששומר את נתיב הקלט לתמונה שהוכנס על ידי המשתמש.
18. Palette – מערך ששומר את פלטת הצבעים בעת קריאת תמונה.
19. ScrLine – מערך ששומר את המידע של השורה הנוכחית בתמונה, אך רק את ערכי האפור.
20. OutputLine – מערך ששומר את כל המידע של השורה הנוכחית בתמונה.
21. currentLine – מערך ששומר את ההמרה של שורת ערכי האפור, לשורת ערכי ascii.
22. Header – מערך ששומר את ה-header של התמונה הנוכחית.
23. output\_file\_handle – משתנה ששומר את אובייקט ה-handle שהתקבל ממערכת ההפעלה לאחר פתיחת קובץ הפלט.
24. error\_file\_handle - משתנה ששומר את אובייקט ה-handle שהתקבל ממערכת ההפעלה לאחר פתיחת קובץ השגיאה.
25. theme\_file\_handle - משתנה ששומר את אובייקט ה-handle שהתקבל ממערכת ההפעלה לאחר פתיחת קובץ תפריט הפתיחה.
26. pathSuffix – הסיומת של נתיב לתמונת bmp.

# טבלת פרוצדורות

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **שם הפרצדורה** | **טענת כניסה** | **טענת יציאה** | **פעולה** | **מקור עזר** |
| \_open\_file | שם הקובץ לפתיחה,  מקום מיועד לשמור את אובייקט הקובץ | פותחת את הקובץ, אם הפתיחה נכשלה, מעבירה ערך החזרה מתאים ל-ax. | פתיחת קובץ | מרכז הסייבר |
| Display\_error | מקום מיועד לשמירת אובייקט של קובץ | מציגה על המסך את תמונת השגיאה | הצגת תמונת שגיאה |  |
| ReadHeader | אובייקט קובץ bmp לקריאה, מערך לשמירת ה-header | קוראת את ה-header מתוך תמונת bmp ושומרת אותו במערך | עוברת מעל ה-header בתמונה | מרכז הסייבר |
| readPalette | אובייקט קובץ bmp לקריאה, מערך לשמירת פלטת הצבעים | קוראת את הפלטה, ושומרת אותה במערך | קריאת פלטת הצבעים מהתמונה | מרכז הסייבר |
| readOutLine | אובייקט קובץ bmp לקריאה, מערך לשמירת השורה | קוראת שורה מתמונת ה-bmp ושומרת אותה במערך | קריאת שורה מתמונה |  |
| CopyPal | מערך שמכיל את פלטת הצבעים | מעתיקה את פלטת הצבעים לזכרון הוידאו במחשב, כך שלאחר מכן הצבעים מומרים לפי הפלטה | הגדרת הצבעים לפי הפלטה | מרכז הסייבר |
| CopyBitMap | אובייקט קובץ bmp לקריאה, מערך לשמירת שורה נוכחית לקריאה | הצגת התמונה על המסך על ידי שליחת צבעי התמונה למסך. ההעתקה מתבצעת בסדר הפוך, כיוון ש-bmp שומר אותה הפוכה. | הצגת התמונה על המסך | מרכז הסייבר |
| createFile | שם הקובץ ליצירה,  אובייקט לשמירת אובייקט הקובץ הנוצר | יצירה של קובץ ושמירת האובייקט המתאים לו | יצירת קובץ |  |
| convertPalToAscii | אובייקט קובץ bmp של תמונת הקלט,  אובייקט קובץ txt לשמירת ערכי ההאסקי,  מערך לשמירת שורה מקובץ הקלט, מערך לשמירת ערכי האסקי לפני כתיבה לקובץ | קוראת את התמונה שורה אחר שורה, וכל שורה ממירה לערכי האסקי לפי ה-ascii table. הערכים נשמרים ונכתבים לקובץ ה-txt. | המרת ערכי אפור בתמונת bmp להאסקי וכתיבתם לקובץ |  |
| closeFile | אובייקט קובץ לסגירה | סגירת קובץ על ידי קריאה למערכת הפעלה | סגירת קובץ | מרכז הסייבר |
| Display\_image | אובייקט קובץ קלט bmp, שם קובץ הקלט, מערך לשמירת שורה לטובת כתיבה למסך, מערך לשמירת פלטת הצבעים, מערך לשמירת ה-header | הצגה של תמונה על המסך על ידי כתיבה לזכרון הוידאו של המחשב, ושינוי פלטת הצבעים שלו, לפי הפלטה של קובץ הקלט. | הצגת תמונה על המסך |  |
| ValidPath | ארבעת הבתים האחרונים של הנתיב שהוכנס על ידי המשתמש | בודקת האם ארבעת הבתים הם הסיומת של נתיב לקובץ bmp | בודקת האם הנתיב לקובץ תקין |  |
| getClientPath | אין | קבלת נתיב לקובץ מהשתמש, ובדיקה האם הוא תקין או לא. נתיב לא תקין – מוצג מסך שגיאה, והמשתמש בוחר האם להכניס נתיב שוב, או לצאת מהתכנית. | קבלת נתיב מהשתמש |  |
| Set\_cursor\_position | אין | מיקום העכבר במיקום שנקבע בראש, על המסך | שינוי מיקום העכבר |  |
| Display\_menu | אין | צובעת את המסך בצבע שהוגדר מראש, מדפיסה את הלוגו של האפליקציה, ואת תפריט הפתיחה. לאחר מכן מבקשת קלט מהמשתמש לנתיב לקובץ. הפעולה ממשיכה, כל עוד הנתיב אינו חוקי. | הצגת התפריט הראשי של התכנית |  |
| Control\_flow | אין | מחכה לקלט מהשתמש במסך הפתיחה של התכנית. אם נלחץ המקש m, מעבירה לתפריט הראשי, אם נלחץ q, התכנית נסגרת. אם המשתמש לחץ עם העכבר על הכפתור השחור בצד השמאלי התחתון של המסך, מעבירה לתפריט הראשי. | בקרת מסך הפתיחה |  |
| Draw\_rect | קוארדינטה אופקית, קוארדינטה אנכית, ממד אורך ורוחב,  צבע | מציירת ריבוע על המסך שמתחיל בקוארדינטות הקלט, בעל צבע לפי צבע הקלט, ואורך לפי אורך הקלט | מציירת ריבוע על המסך |  |
| Draw\_line | קוארדינטה אופקית, קוארדינטה אנכית, ממד אורך,  צבע | מציירת קו ישר על המסך שמתחיל בקוארדינות שמתקבלות, בעל צבע לפי צבע הקלט, ואורך לפי אורך הקלט | מציירת קו ישר על המסך |  |
| Draw\_pixel | קוארדינטה אופקית, קוארדינטה אנכית,  צבע | צובעת פיקסל במיקום הנתון, בצבע הנתון. | צובעת פיקסל במסך |  |
| Display\_theme | נתיב לקובץ תמונה | מציגה את תמונת הפתיחה על המסך, וסוגרת את הקובץ לאחר הפתיחה | הצגת מסך הפתיחה |  |
| Press\_button | קוארדינטה אופקית, קוארדינטה אנכית, ממד אורך | בודקת האם המשתמש לחץ על הריבוע במסך שמוגדר לפי הנקודה הנתונה, והאורך. | בדיקה האם המשתמש לחץ על כפתור הריבוע |  |
| Display\_user\_image | אובייקט קובץ bmp, שם הקובץ,  מערך לשמירת שורה,  מערך לשמירת פלטת הצבעים, מערך לשמירת ה-header | מציגה את התמונה של המסך על ידי מעבר על התמונה ובחירת הבתים המייצגים צבע, כתיבת הפלטה לזכרון הוידאו, והדפסת הצבעים למסך | הצגת תמונת המשתמש על המסך |  |
| Convert\_to\_ascii\_all | אין | ממירה את תמונת המשתמש לתמונת ascii ושומרת את התוצאה בקובץ פלט מסוג txt | ממירת תמונה לתמונת ascii |  |
| start | אין | מבצעת לכל היותר 10 איטרציות של יצירת תמונת האסקי | מתחילה את התכנית |  |

# תרשים זרימה כללי של הפרויקט

Diagram

Description automatically generated

# תרשימי זרימה של פרוצדורות

התכנית מתחילה ב-start.

**טענת כניסה:** אין.

**טענת יציאה:** מריצה את כל קוד התכנית, החל מההתחלה ועד הסוף.

Diagram

Description automatically generatedלהל״ן, תרשים זרימה של הפרוצדורה.

וקטע הקוד המתאים לה, שמריץ 10 איטריציות של המרת תמונה להאסקי והצגתה על המסך:

start:

*mov* ax, @data

*mov* ds, ax

keep\_doing\_till\_quit:

*; convert path*

*mov* al, [number\_of\_files]

*add* al, 48d

*mov* [output\_file\_name], al

*push* *offset* \_image\_name

*call* display\_theme

*push* [color]

*push* [len]

*push* [y\_coordinate]

*push* [x\_coordinate]

*call* draw\_rect

*call* control\_flow

*call* convert\_to\_ascii\_all

*inc* [number\_of\_files]

*mov* al, [number\_of\_files]

*cmp* al, MAX\_FILES

*JE* exit

*jmp* keep\_doing\_till\_quit

exit:

*mov* ax, 4c00h

*int* 21h

END start

בתוך איטרציה זו, אנו מציגים את תפריט המשתמש על ידי הדפסת תמונת הפתיחה על המסך, להל״ן תרשים הזרימה המתאים לפרוצדורה:

Diagram

Description automatically generated

בנוסף, מצורף קטע הקוד הרלוונטי.

**טענת כניסה**: אובייקט handle ששומר את הקובץ הפתוח, שם קובץ הקלט, מערך לשמירת שורה, מערך לשמירת פלטת צבעים, מערך לשמירת ה-header.

**טענת יציאה**: הצגת תמונה על המסך.

*; Docs*

*; display BMP image specified by input file*

*; \* @param input\_handle (arg1) handle*

*; \* @param input\_name (arg2) const char \**

*; \* @param ScrLine (arg3) char \**

*; \* @param Palette (arg4) char \**

*; \* @param Header (arg5) char \**

*; void \_display\_image(Handle \*input\_handle, const char \* input\_name,*

*; char \*ScrLine, char \*Palette, char \*Header)*

*proc* display\_image

*push* bp

*mov* bp, sp

*push* ax

*push* si

*mov* ax, 13h

*int* 10h

*push* arg1

*push* arg2

*call* \_OPEN\_FILE

*push* arg5

*push* arg1

*call* READHEADER

*push* arg4

*push* arg1

*call* READPALETTE

*push* arg4

*call* COPYPAL

*push* arg3

*push* arg1

*call* COPYBITMAP

*pop* si

*pop* ax

*pop* bp

*ret* 10d

*endp* display\_image

Diagram

Description automatically generated לאחר מכן, אנו עוברים לבקרת מסך הפתיחה והתכנית בכלל, שמתבצעת על ידי הפרוצדורה control\_flow, וכאן תוכלו לראות את תרשים הזרימה שלה:

וגם, את הקוד המתאים לה. בעת ריצתה, היא מציירת את העכבר על המסך, ומאפשרת למשתמש להזיז אותו על פני המסך, וללחוץ איתו על הכפתור השחור למטה.

**טענת כניסה**: אין.

**טענת יציאה**: קבלת קלט מהשתמש והעברת המשתמש לרכיב הרלוונטי במערכת.

*; Docs*

*; contorls the program flow*

*; int \_control\_flow()*

*proc* control\_flow

*push* bp

*mov* bp, sp

*pusha*

*mov* ax, 1h

*int* 33h

*; wait for character*

wait\_for\_character:

*mov* ah, 06h

*mov* dl, 0FFh

*int* 21h

*cmp* al, MENU\_CHAR *; menu*

*JE* menu\_case

*cmp* al, QUIT\_CHAR *; quit*

*JE* quit\_case

*push* [len]

*push* [y\_coordinate]

*push* [x\_coordinate]

*call* press\_button

*cmp* ax, PRESSED

*JAE* menu\_case

*JMP* wait\_for\_character *; restrict to m, q*

menu\_case:

*call* display\_menu

*cmp* ax, QUIT\_OPTION

*JE* quit\_case

*jmp* end\_control\_flow

quit\_case:

*mov* ah, 0

*mov* al, 2

*int* 10h

*mov* ax, 4c00h

*int* 21h

end\_control\_flow:

*popa*

*pop* bp

*ret*

*endp* control\_flow

בתוך קטע קוד זה התכנית מבקשת מהמשתמש נתיב לקובץ התמונה אותו הוא רוצה להמיר. הפרוצדורה שמבצעת זאת

היא getclientpath.

תרשים הזרימה שלה מופיע בעמוד הבא:

Diagram

Description automatically generatedלהל״ן קטע הקוד הרלוונטי.

**טענת כניסה**: אין.

**טענת יציאה**: מבקשת מהמשתמש תווים בשביל נתיב לקובץ התמונה בשביל המרה.

במידה שהמשתמש הכניס נתיב לא חוקי, הפרוצדורה תעביר אותו למסך שגיאה, בו הוא יבחר האם לחזור ולהקיש נתיב חוקי, או לצאת מהתכנית.

*; Docs*

*; get image path from user*

*; int \_get\_client\_path()*

*proc* getClientPath

*push* bp

*mov* bp, sp

*push* cx

*push* si

*push* dx

*mov* si, *offset* input\_file\_path

*mov* cx, PATH\_LENGTH

getCharacters:

*; get character*

*mov* ah, 1h

*int* 21h

*cmp* al, 13d *; compare to enter*

*JE* finishedReading

*mov* [*byte* *ptr* si], al

*inc* si

*loop* getcharacters

finishedReading:

*mov* [*byte* *ptr* si], 0d *; null character*

*sub* si, BMP\_END\_SIZE *; last four bytes*

*; mov ah, 02d*

*; mov al, [byte ptr si]*

*; int 21h*

*push* si

*; chack if path ends with .bmp*

*call* validPath

*cmp* ax, VALID\_FILE *; no error*

*JE* continueClientPath

*; load one character*

waitForContCharacters:

*mov* ah, 06h

*mov* dl, 0FFh

*int* 21h

*cmp* al, CONT\_CHAR *; cont*

*JE* finished\_continue

*cmp* al, QUIT\_CHAR *; quit*

*JE* quit\_finished

*JMP* waitForContCharacters *; restrict to m, q*

finished\_continue:

*mov* ax, INVALID\_PATH\_VAL

*jmp* CONTINUECLIENTPATH

quit\_finished:

*mov* ax, QUIT\_OPTION

continueClientPath:

*pop* dx

*pop* si

*pop* cx

*pop* bp

*ret*

*endp* getclientpath

לאחר מכן, התכנית תעבור לשלב המרת התמונה להאסקי.

הפרוצדורה שמבצעת זאת היא convert\_to\_ascii\_all.

להל״ן תרשים הזרימה שלה:

Diagram

Description automatically generatedהפרוצדורה תציג את התמונה על המסך, ולאחר מכך תתחיל בהליך תרגום התמונה להאסקי. כמו כל קריאה של תמונת bmp, היא תקרא את השדות הרלוונטים, רק שבשלב העתקת הפלטה לזכרון הוידאו, היא תבצע משהו שונה, והוא המרה במקום כתיבה לזכרון הוידאו. נצרף את הקוד הרלוונטי:

**טענת כניסה**: אין.

**טענת יציאה**: הצגת תמונת המשתמש על המסך, יצירת קובץ פלט, המרת התמונה למחרוזת האסקי וכתיבת המחרוזת לקובץ הפלט. לבסוף היא סוגרת הקבצים.

*; Docs*

*; convert image fiven by user to ascii and save it in the file "ASCIIGEN<num>.txt"*

*; void \_convert\_to\_ascii\_all()*

*proc* convert\_to\_ascii\_all

*push* bp

*mov* bp, sp

*pusha*

*; push ds*

*; push offset PaletteBuffer*

*; call savepalette*

*push* *offset* Header

*push* *offset* Palette

*push* *offset* ScrLine

*push* *offset* input\_file\_path

*push* *offset* input\_file\_handle

*call* display\_user\_image

*push* *offset* input\_file\_handle

*push* *offset* input\_file\_path

*call* \_open\_file

*push* *offset* output\_file\_handle

*push* *offset* output\_file\_name

*call* createfile

*; process BMP file*

*push* *offset* Header

*push* *offset* input\_file\_handle

*call* ReadHeader

*push* *offset* Palette

*push* *offset* input\_file\_handle

*call* Readpalette

*push* *offset* currentLine

*push* *offset* OutputLine

*push* *offset* output\_file\_handle

*push* *offset* input\_file\_handle

*call* convertPalToAscii

*; wait for key press*

*mov* ah, 1h

*int* 21h

*; back to text mode*

*mov* ah, 0

*mov* al, 2

*int* 10h

*; close output file*

*push* [output\_file\_handle]

*call* closefile

*push* [input\_file\_handle]

*call* closefile

*popa*

*pop* bp

*ret*

*endp* convert\_to\_ascii\_all

Diagram

Description automatically generated את שלב התרגום עצמו, כלומר, המרת הצבעים להאסקי, מבצעת הפרוצדורה convertPalToAscii שלה תרשים הזרימה הבא:

הפרוצדורה עוברת על התמונה שורה שורה מהסוף להתחלה (המידע בתמונת bmp הוא הפוך), ובוחרת את הערך הראשון מבין רביעיית ה-RGB, שכן בתמונת ערכי אפור, השלשה זהה. את תרגום ערך האפור לתו האסקי, התכנית מבצעת לפי האלגוריתם שיוצג בהמשך. להל״ן קטע הקוד הרלוונטי.

**טענת כניסה**: אובייקט handle לשמירת קובץ קלט ה-bmp הפתוח, אובייקט handle לשמירת קובץ הפלט הפתוח, מערך לשמירת שורה בעת קריאת הקובץ, מערך לכתיבת שורה לקובץ הקלט.

**טענת יציאה**: המרת ערכי האפור בתמונה לערכי האסקי, ושמירתם בקובץ הפלט.

*; Docs*

*; convert image specified by file to ascii image*

*; \* @param input\_file\_handle (arg1) handle*

*; \* @param output\_file\_handle (arg2) handle*

*; \* @param output\_line (arg3) string*

*; \* @param current\_line (arg4) string*

*; void \_convert\_pal\_to\_ascii(Handle \*input\_file\_handle, Handle \*output\_file\_handle*

*; char \*output\_line, char \*current\_line)*

*proc* convertPalToAscii

*; BMP graphics are saved upside-down*

*; read the graphic line by line (200 lines in vga format)*

*; displaying the lines from bottom to top*

*; move file cursor to last line*

*push* bp

*mov* bp, sp

*push* cx

*push* dx

*push* ax

*push* bx

*push* si

*push* di

*mov* cx, BMP\_ROWS\_AMOUNT

WriteLines:

*push* cx *; saves cx*

*; set file cursor to next line (from end to start)*

*mov* dx, cx *; compute lines(from end)\*320*

*shl* cx, 6

*shl* dx, 8

*add* dx, cx

*add* dx, 1079d

*mov* ah, 42h

*mov* al, 00h

*mov* si, arg1 *; input file handle*

*mov* bx, [si]

*xor* cx, cx

*int* 21h

*; di =cx\*320, point to the correct screent line*

*push* arg3 *; output\_line*

*push* arg1 *; input file handle*

*call* readoutline

*mov* si, arg3 *; output line*

*mov* di, arg4 *; current line*

*mov* cx, BMP\_COLS\_AMOUNT

convertLineToAscii:

*; mov ax, [si]*

*; convert index pixel\*table\_length / 255*

*xor* ah, ah

*mov* al, [*byte* *ptr* si]

*push* di *; saves current\_line ptr*

*mov* di, *offset* ascii\_table

*add* di, ax

*mov* al, [*byte* *ptr* di]

*pop* di *; restore current line ptr*

*mov* [*byte* *ptr* di], al *; store new ascii value in current line*

*inc* di

*inc* si

*loop* convertlinetoascii

*mov* [*byte* *ptr* di], 10d *; new line*

*; write line to out file*

*mov* ah, 40h

*mov* si, arg2 *; output file handle*

*mov* bx, [si]

*mov* cx, 321

*mov* dx, arg4 *; current line to write*

*int* 21h

*pop* cx *; restore cx*

*loop* WriteLines

*; restore regs*

*pop* di

*pop* si

*pop* bx

*pop* ax

*pop* dx

*pop* cx

*pop* bp

*ret* 8d

*endp* convertPalToAscii

כאן למעשה, מסתכמות, בגדול, הפרוצדורות בתכנית. קיימות כמובן פרוצדורות נוספות, שאחראיות על בדיקת תקינות קלט, פתיחה, סגירה ויצירה של קבצים, ציור מלבן על המסך, ועוד, אך הן פשוטות, ויותר שוליות, מאשר אלה שהוצגו כאן. כמו כן, הן מוצגות בקטע הקוד המלא, בסוף המסמך.

# אלגוריתמים מיוחדים

לתכנית שלושה אלגוריתמים מיוחדים מרכזיים:

1. הצגת התמונה על המסך.
2. הפיכת תמונה לתמונת האסקי.
3. המרה של תמונה לתמונת bmp.
4. אלגוריתם הבקרה של התכנית.

אלגוריתם הבקרה מתואר בתרשים הזרימה של התכנית, ולכן לא יתואר כאן.

## הצגת תמונה על המסך

כדי להיות מסוגלים להציג תמונה על המסך צריך לדעת באיזה פורמט התמונה תגיע. קיימים הרבה סוגי פורמט לשמירת תמונות, הנפוץ ביניהם הוא JPEG ולכן זו מעין בחירה טבעית. הבעיה היא שהוא מבצע דחיסה פנימית של המידע, שכוללת שימוש בטרנספורמציית קוסינוס, ובטרנספורמציית wavelet, מה שדורש דיוק נומרי רב, שקשה להשיג באסמבלי. פורמט אלטרנטיבי, הוא BMP. אמנם הוא לא בשימוש רב, אך הוא מאוד פשוט לעבודה. הוא לא מבצע דחיסה של המידע, הוא מציג אותו באופן סדרתי, ומאוד קל לשלוף אותו. המידע בתמונה מחולק לכמה שדות:

1. Header – פתיחה של הקובץ, והיא בת 54 בתים.
2. Palette – 256 רביעיות. כל רביעייה מייצגת צבע אחד, לפי מרחב הצבע RGB, עם הסייג שהמידע מופיע בסדר הפוך, כלומר BGR. הבית הרביעי הוא הפרדה מהרביעה הבאה. סך הכל בשדה זה 1024 בתים.
3. Data – מכיל את כל הצבעים של הפיקסלים בתמונה, כלומר, מכיל ערכים של 0-255. כל ערך מייצג מיקום בפלטת הצבעים, ולכן על מנת לקבל את הצבע, יש לגשת לפלטה ולשלוף משם את הצבע במקום המתאים. אנו מניחים שגודל התמונה הוא 320 עמודות ו-200 שורות, ולכן יש בשדה זה סך הכל 64,000 בתים. המידע בשדה זה מוצג באופן הפוך, כלומר, יש לקרוא אותו מלמטה למעלה ולכתוב את המידע לזכרון הוידאו, כדי לקבל תמונה עם אורינטציה נכונה.

מכאן, כדי להציג תמונה על המסך, מתקבל האלגוריתם הבא:

1. קרא 54 בתים מהתמונה.
2. קרא 1024 בתים ושמור אותם במערך מתאים. כתוב את המידע בפלטת הצבעים של זכרון הוידאו.
3. קרא את המידע מה-data באופן הפוך, כלומר תתחיל מהסוף, ותקרא שורה אחר שורה. תדפיס כל שורה לזכרון הוידאו.

## הפיכת תמונה לתמונת ASCII

כדי להמיר את התמונה להאסקי, ולשמור את התוצאה בקובץ, צריך לקרוא את התמונה באופן דומה לאלגוריתם הקודם, רק שבמקום לכתוב את המידע לזכרון המחשב, יש להמיר כל שורה לערכי האסקי מתאימים ולשמור את המידע בקובץtxt ולבסוף לסגור את הקבצים.

כדי להמיר את התמונה להאסקי, כך שתתקבל מחרוזת שתראה כמו התמונה המקורית, צריך לשים לב לדבר הבא. ערכי האפור שמייצגים את התמונה הם בטווח 0-255, כך שערכים גדולים מסמנים צבע בהיר (לבן), וערכים נמוכים מסמנים צבע כהה (שחור). לכן, עלינו להתאים ערך גבוה לתו שבקושי נראה על המסך, כדי שיתקבל רקע לבן בעיקר, ולהפך כשמדובר בערך נמוך. היות שאין לנו 256 תווים מתאימים, צריך לבחור רשימה מצומצמת יותר. נביט בהתאמה הבאה:

$@B%8&WM#\*oahkbdpqwmZO0QLCJUYXzcvunxrjft/|()1{}[]?-\_+~<>i!lI;:,"^`\'.

זו התאמה של כ-70 תווים, ונבחין שככל שמתקדמים ברשימה ימינה, התווים נהיה יותר ״בהירים״. נוכל להתאים כל ערך 0-255 לרשימה, כך שיישמר יחס בין הערכים, כלומר ערכים גבוהים יקבלו ערך גבוה יותר ברשימה, וערכים נמוכים, ערך נמוך יותר. כדי לעשות את זה, ננרמל את הטווח, על ידי חלוקה ב-255, וקבלת טווח 0-1. לאחר מכן, על ידי כפל בערך הפיקסל, נקבל ערך בין 0-255, שהוא לא בהכרח שלם, לכן ניקח ממנו את הערך השלם העליון. נממש את זה בפייתון:

chars *=* '&@B%8&WM#\*oahkbdpqwmZO0QLCJUYXzcvunxrjft/|()1{}[]?-\_+~<>i!lI;:,"^`\`. '

new\_pixels *=* [chars[math.ceil((len(chars)*-* 1) *\** pixel */* 255)]

*for* pixel *in* pixels]

new\_pixels *=* ''.join(new\_pixels)

ונקבל את טבלת המרה מלאה של ערכי אפור לתווים, אשר שמורה בקובץ menu.asm.

## המרה של תמונה לפורמט bmp

חלק זה התכנית לא מבצעת, אלא המשתמש מבצע לפני שמפעיל אותה. כדי לעשות זאת, אפשר לקחת תמונה בכל פורמט שהוא, ולהמיר אותו ל-bmp [באתר הבא](https://convertio.co/jpg-bmp/). לאחר מכן, יש להמיר אותה למימדים 200 על 320, ולהפוך אותה לתמונת ערכי אפור. לא מצאתי אתר שעושה את זה טוב, אז כתבתי קוד פייתון שמבצע זאת, :

*import* sys

*from* PIL *import* Image

*from* matplotlib.pyplot *import* imread

*from* numpy *import* save

*def* *save\_bmp\_to\_gray*():

image\_path *=* sys.argv[1]

img *=* Image.open(image\_path)

*# resize the image*

img *=* img.convert('L')

*# img.save("grayscale.bmp")*

*# width, height = img.size*

img *=* img.resize((320, 200))

*# new size of image*

*# print(img.size)*

img.save("temp.bmp")

*if* \_\_name\_\_ *==* "\_\_main\_\_":

save\_bmp\_to\_gray()

קובץ זה ניתן להריץ באמצעות הפקודה:

python3 convert.py <path\_to\_bmp\_file>

כאשר יש להחליף את החלק של ה-<> בנתיב לקובץ ה-bmp. צירפתי את הקובץ לתיקייה הפרוייקט, ושמו convert.py.

# סיכום ורפלקציה

כשהתחלתי לעבוד על הפרוייקט, רציתי לממש שיטת הצפנה ידועה, המכונה RSA. למדתי עליה בקפידה באוניברסיטה, וחשבתי שיהיה מעניין לממש אותה. היות שהיא מורכבת, מימשתי אותה קודם בשפת c, כדי שיהיה לי קל להמיר אותו לאסמבלי, במחשבה שכאשר האלגוריתמים עובדים טוב, כל מה שנשאר זה לממש. למעשה, כדי להיות כמה שיותר מסודר, כתבתי קוד בשפת פייתון שממיר את קוד ה-c שכתבתי, **לחתימות** של פרוצדורות באסמבלי, כך שיהיה לי ממש api כתוב באסמבלי. הקוד עבד נהדר ב-c, אבל בניסיוני לתרגם אותו לאסמבלי, נתקלתי בכמה בעיות מרכזיות, הראשונה היא שהסטאק שמוקצה לתכנית, באופן מפתיע, קרס מאוד מהר. הדבר הגיוני, כיוון שהתכנית משתמשת בהרבה מאוד משתנים לוקאליים כבדים. הגדלתי את הסטאק, והדבר חזר על עצמו לאחר מכן, יתכן שהיה באג במימוש, אך הבנתי שיתכן שמדובר במשימה מורכבת מדי.[[1]](#footnote-1)

עקב כך, החלטתי לעבור לתחום אחר שאני אוהב – עיבוד תמונה, והמרת תמונה להאסקי בפרט. תחילה, לא היה לי ברור איך להתחיל, שכן כיצד אמורים לקרוא תמונה באסמבלי? חקרתי בגוגל, ומצאתי את [הספר](https://data.cyber.org.il/assembly/assembly_book.pdf) של מרכז הסייבר, ובו דוגמאות רבות לאיך אפשר לעבוד עם תמונות, והדבר מאוד עזר לי. ההסברים כללו כיצד אפשר לגשת לזכרון הוידאו במחשב, ולכתוב ישירות לפורטים את המידע, מה שעזר לי להבין את התאוריה מאחורי העיסוק עם התמונה והצגתה במסך. אמנם, בהתחלה, לא זכרתי בדיוק איך לקרוא לכל פסיקה, אך עם הזמן התרגלתי, ומצאתי [אתר נהדר](https://stanislavs.org/helppc/) שמסביר על כל פסיקה במפורט, שבאמצעותו מצאתי פסיקות שימושיות רבות. על כן, ברצוני להודות לכותבי ספר מרכז הסייבר וליוצרי האתר.

אחד האתגרים המרכזיים שצצו, היה כיצד להמיר כל ערך אפור לתו האסקי. מצאתי אמנם נוסחא להמרה, אך היא דרשה שימוש במספרים ממשיים, מה שקשה לבצע באסמבלי. הפתרון שמצאתי היה להגדיר טבלא של תווים מראש, כלומר, הרצתי קוד בשפה עילית (בחרתי ב-python) שיצר את הטבלא, שם אפשר להשתמש במספרים ממשיים בקלות, כך שכל שהייתי צריך לעשות זה לגשת לטבלא לפי הערך האפור, ולקחת את התו.

המסקנות שעלו לי בסוף כתיבת הקוד, הן שכדאי לתכנן הכל מראש, לפי תחילת הכתיבה. כלומר, להתחיל מרעיון, ולבדוק שריאלי לממש אותו באסמבלי, שכן יש הרבה רעיונות טובים, אבל חלקם קשים מאוד למימוש. כמו כן, אני ממליץ להעזר בשפות עיליות לכתיבת כלי עזר, כמו שביצעתי עם טבלת ההמרה, או יצירת ה-api, הדבר חוסך המון זמן, וגם מהנה.

לסיכום, מאוד נהנתי בעת כתיבת הפרוייקט, ואני ממליץ לתלמידים בחום להשתמש במקורות המידע שציינתי לעיל, ולחקור על כמה שיותר פסיקות, שכן, הרבה פעמים הדבר יכול מאוד להועיל.

# קוד הפרויקט

## קובץ הקבועים menu.asm

הקובץ מכיל קבועים חשובים. הוא מיובא על ידי base.asm.

menu *db* '`##::::`##:`########:`##::: ##:`##::::`##:::::`######:::`########:`##::: ##:', 13, 10

*db* ' ###::`###: ##.....:: ###:: ##: ##:::: ##::::`##... ##:: ##.....:: ###:: ##:', 13, 10

*db* ' ####`####: ##::::::: ####: ##: ##:::: ##:::: ##:::..::: ##::::::: ####: ##:', 13, 10

*db* ' ## ### ##: ######::: ## ## ##: ##:::: ##:::: ##::`####: ######::: ## ## ##:', 13, 10

*db* ' ##. #: ##: ##...:::: ##. ####: ##:::: ##:::: ##::: ##:: ##...:::: ##. ####:', 13, 10

*db* ' ##:.:: ##: ##::::::: ##:. ###: ##:::: ##:::: ##::: ##:: ##::::::: ##:. ###:', 13, 10

*db* ' ##:::: ##: ########: ##::. ##:. #######:::::. ######::: ########: ##::. ##:', 13, 10

*db* '..:::::..::........::..::::..:::.......:::::::......::::........::..::::..::', 13, 10

*db* '################################################################################'

*db* '#### THE ASCII GEN ####'

*db* '#### SYSTEM ####'

*db* '################################################################################', 13, 10, '$'

ascii\_table *db* '&@@@BBBB%%'

*db* '%%888&&&&W'

*db* 'WWWMMM####'

*db* '\*\*\*\*oooaaa'

*db* 'ahhhhkkkkb'

*db* 'bbddddpppp'

*db* 'qqqwwwwmmm'

*db* 'mZZZOOOO00'

*db* '00QQQQLLLC'

*db* 'CCCJJJJUUU'

*db* 'YYYYXXXXzz'

*db* 'zccccvvvvu'

*db* 'uunnnnxxxx'

*db* 'rrrrjjjfff'

*db* 'ftttt///||'

*db* '||(((()))1'

*db* '111{{{{}}}'

*db* '}[[[]]]]\_\_'

*db* '\_\_---\_\_\_\_+'

*db* '+++~~~<<<<'

*db* '>>>>iii!!!'

*db* '!llllIIII;'

*db* ';;::::,,,,'

*db* '..........'

*db* '..........'

*db* '. '

## הקובץ הראשי base.asm

הקובץ מכיל את כל פרוצדורות הפרוייקט.

.286

IDEAL

MODEL small

STACK 9999h

DATASEG

include "menu.asm"

*; constant macros*

PRESSED *equ* 1d

NOT\_PRESSED *equ* 0d

MENU\_CHAR *equ* 'm'

QUIT\_CHAR *equ* 'q'

QUIT\_OPTION *equ* 2d

MENU\_COLOR *equ* 031H

START\_MENU\_ROW *equ* 2d

START\_MENU\_COL *equ* 0d

INVALID\_PATH\_VAL *equ* 1d

CONT\_CHAR *equ* 'c'

VALID\_FILE *equ* 0d

BMP\_END\_SIZE *equ* 4d

BMP\_COLS\_AMOUNT *equ* 320d

BMP\_ROWS\_AMOUNT *equ* 200d

BMP\_HEADER\_SIZE *equ* 54d

BMP\_PALETTE\_SIZE *equ* 400h *; 256\*4*

SCREEN\_START\_OFFSET *equ* 0A000h

PATH\_LENGTH *equ* 100d

OUTPUT\_PATH\_LENGTH *equ* 14d

MAX\_FILES *equ* 10d

*; function macros*

arg1 *equ* [*word* *ptr* bp*+*4]

arg2 *equ* [*word* *ptr* bp*+*6]

arg3 *equ* [*word* *ptr* bp*+*8]

arg4 *equ* [*word* *ptr* bp*+*10]

arg5 *equ* [*word* *ptr* bp*+*12]

*; variables*

NUMBER\_OF\_FILES *db* 0d

PaletteBuffer *db* 300h *dup*(0)

\_image\_name *db* 'ascii.bmp',0

\_error\_image *db* 'Error.bmp', 0

\_open\_menu *db* 9d, 9d, 'Welcome! This is the ascii image generator system!',13 ,10, 13 ,10

*db* 9d, 9d, 'Please Enter the path for your image file.', 13, 10, 13 ,10

*db* 9d, 9d,'It has to be in BMP format 200x320.', 13, 10, 13 ,10

*db* 9d, 9d,'Enter path (Maximum 100 characters): ', '$'

output\_file\_name *db* 0, 'asciiGenResult.txt', 0 *; 30d dup(0)*

len *dw* 20

y\_coordinate *dw* 180

y\_temp *dw* 0

x\_coordinate *dw* 0

x\_temp *dw* 0

color *dw* 60d

CURRENT\_LINE\_LENGTH *equ* 321d

input\_file\_handle *dw* ?

input\_file\_path *db* 101 *dup*(0)

Palette *db* BMP\_PALETTE\_SIZE *dup*(0)

ScrLine *db* BMP\_COLS\_AMOUNT *dup*(0)

OutputLine *db* 1280 *dup*(0)

currentLine *db* CURRENT\_LINE\_LENGTH *dup*(0)

Header *db* BMP\_HEADER\_SIZE *dup*(0)

output\_file\_handle *dw* ?

error\_file\_handle *dw* ?

theme\_file\_handle *dw* ?

pathSuffix *db* '.bmp', 0

CODESEG

*; Docs*

*; \* open file spesified by filename*

*; \* @param filename (arg1) string*

*; \* @param file handle (arg2)*

*; void \_open\_file(const char \*filename, Handle \*file\_handle)*

*proc* \_open\_file

*push* bp

*mov* bp, sp

*push* dx

*push* si

*mov* si, arg2 *; store handle object address*

*mov* ah, 3Dh

*xor* al, al

*mov* dx, arg1 *; push file name*

*int* 21h

*jc* openerror *; check for errors*

*mov* [si], ax

*mov* ax, VALID\_FILE

*jmp* opened\_sucessfuly

openerror:

*; display error theme*

*mov* ax, INVALID\_PATH\_VAL

opened\_sucessfuly:

*; restore registers*

*pop* si

*pop* dx

*pop* bp

*ret* 4d

*endp* \_open\_file

*; Docs*

*; display BMP error image*

*; \* @param input\_handle (arg1) handle*

*; void \_display\_error(Handle \*input\_handle)*

*proc* display\_error

*push* bp

*mov* bp, sp

*push* si

*push* *offset* Header

*push* *offset* Palette

*push* *offset* ScrLine

*push* arg1

*push* *offset* error\_file\_handle

*call* display\_image

*push* [error\_file\_handle]

*call* closefile

*pop* si

*pop* bp

*ret* 2d

*endp* display\_error

*; Docs*

*; \* Read BMP file's header*

*; \* @param input\_file\_handle (arg1) string*

*; \* @param header (arg2) string*

*; void \_read\_header(Handle \*input\_file\_handle, char \*header)*

*proc* ReadHeader

*; read bmp file header, 54 bytes*

*push* bp

*mov* bp, sp

*push* ax

*push* bx

*push* cx

*push* dx

*push* si

*mov* si, arg1 *; input file handle*

*mov* ah, 3fh *; read using handle*

*mov* bx, [si]

*mov* cx, BMP\_HEADER\_SIZE

*mov* dx, arg2

*int* 21h

*; restore regs*

*pop* si

*pop* dx

*pop* cx

*pop* bx

*pop* ax

*pop* bp

*ret* 4d

*endp* ReadHeader

*; Docs*

*; \* Read BMP file's palette*

*; \* @param input\_file\_handle (arg1) string*

*; \* @param palette (arg2) string*

*; void \_read\_palette(Handle \*input\_file\_handle, char \*palette)*

*proc* ReadPalette

*; read BMP file color palette, 256 colors \* 4 bytes (400h)*

*push* bp

*mov* bp, sp

*push* ax

*push* cx

*push* dx

*push* si

*push* bx

*mov* ah, 3fh

*mov* cx, BMP\_PALETTE\_SIZE

*mov* si, arg1

*mov* bx, [si] *; file handle*

*mov* dx, arg2 *; palette*

*int* 21h

*; restore*

*pop* bx

*pop* si

*pop* dx

*pop* cx

*pop* ax

*pop* bp

*ret* 4d

*endp* ReadPalette

*; Docs*

*; \* Read BMP file's line*

*; \* @param input\_file\_handle (arg1) string*

*; \* @param line (arg2) string*

*; void \_read\_out\_line(Handle \*input\_file\_handle, char \*line)*

*proc* ReadOutLine

*push* bp

*mov* bp, sp

*push* ax

*push* cx

*push* dx

*push* si

*push* bx

*mov* si, arg1 *; input file handle*

*mov* ah, 3fh

*mov* cx, BMP\_COLS\_AMOUNT

*mov* bx, [si]

*mov* dx, arg2 *; address of line*

*int* 21h

*; restore*

*pop* bx

*pop* si

*pop* dx

*pop* cx

*pop* ax

*pop* bp

*ret* 4d

*endp* ReadOutLine

*; Docs*

*; \* copy the colors palette to the video memory*

*; \* The number of the first color should be sent to port 3C8h*

*; \* The palette is sent to port 3C9h*

*; \* copy BMP file's palette to video memory*

*; \* @param palette (arg1) string*

*; void \_copy\_pal(const char \*palette)*

*proc* CopyPal

*push* bp

*mov* bp, sp

*push* si

*push* cx

*push* dx

*push* ax

*mov* si, arg1 *; palette address*

*mov* cx, 256

*mov* dx, 3C8h

*mov* al, 0

*; copy starting color to port 3C8h*

*out* dx, al

*; copy palette itself to port 3C9h*

*inc* dx

PalLoop:

*; Note: colors in a BMP file are saved as BGR values rather than RGB*

*mov* al, [si*+*2] *; get red value*

*shr* al, 2 *; Max. is 255, but vidro palette maximal value is 63,*

*; therefore dividing by 4*

*out* dx, al *; send it*

*mov* al, [si*+*1] *; get green value*

*shr* al, 2

*out* dx, al *; send it*

*mov* al, [si] *; get blue value*

*shr* al, 2

*out* dx, al *; send it*

*add* si, 4 *; point to next color (There is a null chr, after every color, [si+3])*

*loop* palloop

*pop* ax

*pop* dx

*pop* cx

*pop* si

*pop* bp

*ret* 2d

*endp* CopyPal

*; Docs*

*; \* copy image colors to the screen (each color value to the corresponding pixel)*

*; \* BMP graphics are saved upside-down*

*; \* read the graphic line by line (200 lines in vga format)*

*; \* displaying the lines from bottom to top*

*; \* @param input\_file\_handle (arg1) handle*

*; \* @param scrLine (arg2) line*

*; void \_copy\_bit\_map(Handle \*input\_file\_handle, char \*scrLine)*

*proc* CopyBitmap

*; BMP graphics are saved upside-down*

*; read the graphic line by line (200 lines in vga format)*

*; displaying the lines from bottom to top*

*push* bp

*mov* bp, sp

*push* ax

*push* es

*push* cx

*push* di

*push* si

*push* dx

*push* bx

*mov* ax, SCREEN\_START\_OFFSET *; screen start*

*mov* es, ax

*mov* cx, BMP\_ROWS\_AMOUNT

PrintBMPLoop:

*push* cx

*; di =cx\*320, point to the correct screent line*

*mov* di, cx

*shl* cx, 6

*shl* di, 8

*add* di, cx

*; Read one line, load line to ScrLine*

*mov* ah, 3fh *; read from handle object*

*mov* cx, BMP\_COLS\_AMOUNT

*mov* si, arg1

*mov* bx, [si] *; file handle*

*mov* dx, arg2 *; line segment*

*int* 21h

*; Copy one line into video memory*

*cld* *; Clear diection flag, for movsb*

*mov* cx, BMP\_COLS\_AMOUNT

*mov* si, arg2

*rep* *movsb* *; Cope line to the screen*

*; rep movsb is name as the follwing code:*

*; mov es:di, ds:si*

*; inc si*

*; inc di*

*; dec cx*

*; ...;loop until cx=0*

*pop* cx

*loop* PRINTBMPLOOP

*; restore*

*pop* bx

*pop* dx

*pop* si

*pop* di

*pop* cx

*pop* es

*pop* ax

*pop* bp

*ret* 4d

*endp* CopyBitmap

*; Docs*

*; create new file*

*; \* @param output\_file\_name (arg1) string*

*; \* @param output\_file\_handle (arg2) handle*

*; void \_create\_file(const char \*output\_file\_name, Handle \*output\_file\_handle)*

*proc* CreateFile

*push* bp

*mov* bp, sp

*push* ax

*push* cx

*push* dx

*push* si

*mov* si, arg2*; output file handle address*

*mov* ah, 3ch

*mov* cx, 1

*mov* dx, arg1 *; output\_file name*

*int* 21h

*jc* createerror

*mov* [si], ax

*jmp* created\_succesfuly

createError:

*push* *offset* \_error\_image

*call* display\_error

*mov* ah, 1

*int* 21h

*; back to text mode*

*mov* ah, 0

*mov* al, 2

*int* 10h

*mov* dx, *offset* \_open\_menu

*mov* ah, 09h

*int* 21h

*; exit*

*mov* ax, 4c00h

*int* 21h

created\_succesfuly:

*pop* si

*pop* dx

*pop* cx

*pop* ax

*pop* bp

*ret* 4d

*endp* CreateFile

*; Docs*

*; convert image specified by file to ascii image*

*; \* @param input\_file\_handle (arg1) handle*

*; \* @param output\_file\_handle (arg2) handle*

*; \* @param output\_line (arg3) string*

*; \* @param current\_line (arg4) string*

*; void \_convert\_pal\_to\_ascii(Handle \*input\_file\_handle, Handle \*output\_file\_handle*

*; char \*output\_line, char \*current\_line)*

*proc* convertPalToAscii

*; BMP graphics are saved upside-down*

*; read the graphic line by line (200 lines in vga format)*

*; displaying the lines from bottom to top*

*; move file cursor to last line*

*push* bp

*mov* bp, sp

*push* cx

*push* dx

*push* ax

*push* bx

*push* si

*push* di

*mov* cx, BMP\_ROWS\_AMOUNT

WriteLines:

*push* cx *; saves cx*

*; set file cursor to next line (from end to start)*

*mov* dx, cx *; compute lines(from end)\*320*

*shl* cx, 6

*shl* dx, 8

*add* dx, cx

*add* dx, 1079d

*mov* ah, 42h

*mov* al, 00h

*mov* si, arg1 *; input file handle*

*mov* bx, [si]

*xor* cx, cx

*int* 21h

*; di =cx\*320, point to the correct screent line*

*push* arg3 *; output\_line*

*push* arg1 *; input file handle*

*call* readoutline

*mov* si, arg3 *; output line*

*mov* di, arg4 *; current line*

*mov* cx, BMP\_COLS\_AMOUNT

convertLineToAscii:

*; mov ax, [si]*

*; convert index pixel\*table\_length / 255*

*xor* ah, ah

*mov* al, [*byte* *ptr* si]

*push* di *; saves current\_line ptr*

*mov* di, *offset* ascii\_table

*add* di, ax

*mov* al, [*byte* *ptr* di]

*pop* di *; restore current line ptr*

*mov* [*byte* *ptr* di], al *; store new ascii value in current line*

*inc* di

*inc* si

*loop* convertlinetoascii

*mov* [*byte* *ptr* di], 10d *; new line*

*; write line to out file*

*mov* ah, 40h

*mov* si, arg2 *; output file handle*

*mov* bx, [si]

*mov* cx, 321

*mov* dx, arg4 *; current line to write*

*int* 21h

*pop* cx *; restore cx*

*loop* WriteLines

*; restore regs*

*pop* di

*pop* si

*pop* bx

*pop* ax

*pop* dx

*pop* cx

*pop* bp

*ret* 8d

*endp* convertPalToAscii

*; Docs*

*; close file*

*; \* @param file\_handle (arg1) handle*

*; void \_close(Handle \*file\_handle)*

*proc* closeFile

*; saves regs*

*push* bp

*mov* bp, sp

*push* ax

*push* bx

*mov* ah, 3Eh

*mov* bx, arg1 *; file handle*

*int* 21h

*; restore regs*

*pop* bx

*pop* ax

*pop* bp

*ret* 2d

*endp* closeFile

*; Docs*

*; display BMP image specified by input file*

*; \* @param input\_handle (arg1) handle*

*; \* @param input\_name (arg2) const char \**

*; \* @param ScrLine (arg3) char \**

*; \* @param Palette (arg4) char \**

*; \* @param Header (arg5) char \**

*; void \_display\_image(Handle \*input\_handle, const char \* input\_name,*

*; char \*ScrLine, char \*Palette, char \*Header)*

*proc* display\_image

*push* bp

*mov* bp, sp

*push* ax

*push* si

*mov* ax, 13h

*int* 10h

*push* arg1

*push* arg2

*call* \_OPEN\_FILE

*push* arg5

*push* arg1

*call* READHEADER

*push* arg4

*push* arg1

*call* READPALETTE

*push* arg4

*call* COPYPAL

*push* arg3

*push* arg1

*call* COPYBITMAP

*pop* si

*pop* ax

*pop* bp

*ret* 10d

*endp* display\_image

*; Docs*

*; checks if the path given by the user if valid*

*; \* @param last\_four\_bytes (arg1) handle*

*; void \_valid\_path(const char \*last\_four\_bytes)*

*proc* validPath

*push* bp

*mov* bp, sp

*push* si

*push* di

*push* cx

*mov* si, arg1 *; last four bytes of path*

*mov* di, *offset* pathsuffix

*mov* cx, BMP\_END\_SIZE

compareBytes:

*mov* dl, [*byte* *ptr* di]

*cmp* dl, [*byte* *ptr* si]

*JNE* invalidPath

*inc* di

*inc* si

*loop* comparebytes

*jmp* IsvalidPath

invalidPath:

*push* *offset* \_error\_image

*call* display\_error

*mov* ax, INVALID\_PATH\_VAL

*jmp* endValidPath

IsvalidPath:

*mov* ax, VALID\_FILE

*; check file could be opened*

*push* *offset* input\_file\_handle

*push* *offset* input\_file\_path

*call* \_open\_file

*cmp* ax, INVALID\_PATH\_VAL

*je* invalidpath

endValidPath:

*pop* cx

*pop* di

*pop* si

*pop* bp

*ret* 2d

*endp* validPath

*; Docs*

*; get image path from user*

*; int \_get\_client\_path()*

*proc* getClientPath

*push* bp

*mov* bp, sp

*push* cx

*push* si

*push* dx

*mov* si, *offset* input\_file\_path

*mov* cx, PATH\_LENGTH

getCharacters:

*; get character*

*mov* ah, 1h

*int* 21h

*cmp* al, 13d *; compare to enter*

*JE* finishedReading

*mov* [*byte* *ptr* si], al

*inc* si

*loop* getcharacters

finishedReading:

*mov* [*byte* *ptr* si], 0d *; null character*

*sub* si, BMP\_END\_SIZE *; last four bytes*

*; mov ah, 02d*

*; mov al, [byte ptr si]*

*; int 21h*

*push* si

*; chack if path ends with .bmp*

*call* validPath

*cmp* ax, VALID\_FILE *; no error*

*JE* continueClientPath

*; load one character*

waitForContCharacters:

*mov* ah, 06h

*mov* dl, 0FFh

*int* 21h

*cmp* al, CONT\_CHAR *; cont*

*JE* finished\_continue

*cmp* al, QUIT\_CHAR *; quit*

*JE* quit\_finished

*JMP* waitForContCharacters *; restrict to m, q*

finished\_continue:

*mov* ax, INVALID\_PATH\_VAL

*jmp* CONTINUECLIENTPATH

quit\_finished:

*mov* ax, QUIT\_OPTION

continueClientPath:

*pop* dx

*pop* si

*pop* cx

*pop* bp

*ret*

*endp* getclientpath

*; Docs*

*; sets cursor position*

*; int \_set\_cursor\_pos()*

*proc* set\_cursor\_pos

*push* bp

*mov* bp, sp

*push* ax

*push* bx

*mov* ah, 02

*mov* bh, 0 *; page number (0 for graphics modes)*

*; dh = row*

*; dl = col*

*int* 10h

*pop* bx

*pop* ax

*pop* bp

*ret*

*endp* set\_cursor\_pos

*; Docs*

*; displays the program menu*

*; int \_display\_menu()*

*proc* display\_menu

*push* bp

*mov* bp, sp

*; back to text mode*

doTillValidPathOrQuit:

*mov* ah, 0

*mov* al, 2

*int* 10h

*; change bkacground color*

*mov* ah, 00

*mov* al, 03

*int* 10h

*; set color*

*mov* ah, 09 *; DISPLAY OPTION*

*mov* bh, 00 *; PAGE 0*

*mov* al, 20H *; ASCII FOR SPACE*

*mov* cx, 800H *; REPEAT IT 800H, to fill screen*

*mov* bl, MENU\_COLOR *; COLOR*

*int* 10H

*; set cursor position*

*mov* dh, START\_MENU\_ROW *; row*

*mov* dl, START\_MENU\_COL *; col*

*call* set\_cursor\_pos

*mov* dx, *offset* menu

*mov* ah, 09h

*int* 21h

*; print menu string*

*mov* dx, *offset* \_open\_menu

*mov* ah, 09h

*int* 21h

*; get client path and validate*

*call* getclientpath

*cmp* ax, INVALID\_PATH\_VAL

*JE* dotillvalidpathorquit

*pop* bp

*ret*

*endp* display\_menu

*; Docs*

*; contorls the program flow*

*; int \_control\_flow()*

*proc* control\_flow

*push* bp

*mov* bp, sp

*pusha*

*mov* ax, 1h

*int* 33h

*; wait for character*

wait\_for\_character:

*mov* ah, 06h

*mov* dl, 0FFh

*int* 21h

*cmp* al, MENU\_CHAR *; menu*

*JE* menu\_case

*cmp* al, QUIT\_CHAR *; quit*

*JE* quit\_case

*push* [len]

*push* [y\_coordinate]

*push* [x\_coordinate]

*call* press\_button

*cmp* ax, PRESSED

*JAE* menu\_case

*JMP* wait\_for\_character *; restrict to m, q*

menu\_case:

*call* display\_menu

*cmp* ax, QUIT\_OPTION

*JE* quit\_case

*jmp* end\_control\_flow

quit\_case:

*mov* ah, 0

*mov* al, 2

*int* 10h

*mov* ax, 4c00h

*int* 21h

end\_control\_flow:

*popa*

*pop* bp

*ret*

*endp* control\_flow

*; Docs*

*; draws a rectangle*

*; \* @param x\_coordinate (arg1) int*

*; \* @param y\_coordinate (arg2) int*

*; \* @param len (arg3) int*

*; \* @param color (arg4) int*

*; int \_draw\_rect(int x\_coordinate, int y\_coordinate, int len, int color)*

*proc* draw\_rect

*push* bp

*mov* bp, sp

*pusha*

*mov* cx, arg3 *; len*

*mov* ax, arg2 *; y*

*mov* [y\_temp], ax

rect:

*push* arg4

*push* arg3

*push* [y\_temp]

*push* arg1

*call* draw\_line

*inc* [y\_temp] *;column*

*loop* rect

*popa*

*pop* bp

*ret* 8d

*endp* draw\_rect

*; Docs*

*; draws a line*

*; \* @param x\_coordinate (arg1) int*

*; \* @param y\_coordinate (arg2) int*

*; \* @param len (arg3) int*

*; \* @param color (arg4) int*

*; int \_draw\_line(int x\_coordinate, int y\_coordinate, int len, int color)*

*proc* draw\_line

*push* bp

*mov* bp, sp

*pusha*

*; move x\_coordinate to x\_temp*

*mov* ax, arg1 *; x*

*mov* [x\_temp], ax

*mov* cx, arg3

draw:

*; push color, x, y*

*push* arg4 *; color*

*push* arg2 *; y*

*push* [x\_temp] *; x*

*call* draw\_pixel

*inc* [x\_temp]

*loop* draw

*popa*

*pop* bp

*ret* 8d

*endp* draw\_line

*; Docs*

*; draws a pixel in a position*

*; \* @param x\_coordinate (arg1) int*

*; \* @param y\_coordinate (arg2) int*

*; \* @param color (arg3) int*

*; int \_draw\_pixel(int x\_coordinate, int y\_coordinate, int color)*

*proc* draw\_pixel

*push* bp

*mov* bp, sp

*pusha*

*xor* bh, bh *; bh = 0*

*mov* cx, arg1

*mov* dx, arg2

*mov* ax, arg3

*mov* ah, 0ch

*int* 10h

*popa*

*pop* bp

*ret* 6d

*endp* draw\_pixel

*; Docs*

*; displays the main theme*

*; \* @param image\_path (arg1) const char \**

*; void \_display\_theme(const char \*image\_path)*

*proc* display\_theme

*push* bp

*mov* bp, sp

*push* si

*push* *offset* Header

*push* *offset* Palette

*push* *offset* ScrLine

*push* arg1 *; image path*

*push* *offset* theme\_file\_handle

*call* display\_image

*push* [theme\_file\_handle]

*call* closefile

*pop* si

*pop* bp

*ret* 2d

*endp* display\_theme

*; Docs*

*; checks if the user pressed the mouse buttons, and if the cursor was in the square*

*; [arg1, arg1+len] X [arg2, arg2 + len] (cartesean multiplication)*

*; \* @param x\_coordinate (arg1) int*

*; \* @param y\_coordinate (arg2) int*

*; \* @param length (arg3) int*

*; \* @return ax stores 1 if it is true, 0 otherwise*

*; int \_press\_button(int x\_coordinate, int y\_coordinate, int len)*

*proc* press\_button

*push* bp

*mov* bp, sp

*xor* ax, ax

*mov* ax, 3h *; read mouse status and position*

*int* 33h

*mov* ax, NOT\_PRESSED

*cmp* bx, PRESSED *; check left mouse click*

*JNE* end\_press\_button*; if left click not pressed….*

*; compare x*

*push* ax

*mov* ax, arg1

*cmp* cx, ax *; column*

*pop* ax

*JL* end\_press\_button

*push* ax

*mov* ax, arg1

*add* ax, arg3

*cmp* cx, ax

*pop* ax

*JA* end\_press\_button

*; compare y*

*push* ax

*mov* ax, arg2

*cmp* dx, ax *; row*

*pop* ax

*JL* end\_press\_button

*push* ax

*mov* ax, arg2

*add* ax, arg3

*cmp* dx, ax

*pop* ax

*JA* end\_press\_button

*mov* ax, PRESSED

end\_press\_button:

*pop* bp

*ret* 6d

*endp* press\_button

*; Docs*

*; display BMP image specified by input file, but doesn't open file in the begining*

*; \* @param input\_handle (arg1) handle*

*; \* @param input\_name (arg2) const char \**

*; \* @param ScrLine (arg3) char \**

*; \* @param Palette (arg4) char \**

*; \* @param Header (arg5) char \**

*; void \_display\_user\_image(Handle \*input\_handle, const char \* input\_name,*

*; char \*ScrLine, char \*Palette, char \*Header)*

*proc* display\_user\_image

*push* bp

*mov* bp, sp

*mov* ax, 13h

*int* 10h

*push* arg5

*push* arg1

*call* READHEADER

*push* arg4

*push* arg1

*call* READPALETTE

*push* arg4

*call* COPYPAL

*push* arg3

*push* arg1

*call* COPYBITMAP

*mov* si, arg1

*push* [si]

*call* closefile

*pop* bp

*ret* 10d

*endp* display\_user\_image

*; Docs*

*; convert image fiven by user to ascii and save it in the file "ASCIIGEN<num>.txt"*

*; void \_convert\_to\_ascii\_all()*

*proc* convert\_to\_ascii\_all

*push* bp

*mov* bp, sp

*pusha*

*; push ds*

*; push offset PaletteBuffer*

*; call savepalette*

*push* *offset* Header

*push* *offset* Palette

*push* *offset* ScrLine

*push* *offset* input\_file\_path

*push* *offset* input\_file\_handle

*call* display\_user\_image

*push* *offset* input\_file\_handle

*push* *offset* input\_file\_path

*call* \_open\_file

*push* *offset* output\_file\_handle

*push* *offset* output\_file\_name

*call* createfile

*; process BMP file*

*push* *offset* Header

*push* *offset* input\_file\_handle

*call* ReadHeader

*push* *offset* Palette

*push* *offset* input\_file\_handle

*call* Readpalette

*push* *offset* currentLine

*push* *offset* OutputLine

*push* *offset* output\_file\_handle

*push* *offset* input\_file\_handle

*call* convertPalToAscii

*; wait for key press*

*mov* ah, 1h

*int* 21h

*; back to text mode*

*mov* ah, 0

*mov* al, 2

*int* 10h

*; close output file*

*push* [output\_file\_handle]

*call* closefile

*push* [input\_file\_handle]

*call* closefile

*popa*

*pop* bp

*ret*

*endp* convert\_to\_ascii\_all

start:

*mov* ax, @data

*mov* ds, ax

keep\_doing\_till\_quit:

*; convert path*

*mov* al, [number\_of\_files]

*add* al, 48d

*mov* [output\_file\_name], al

*push* *offset* \_image\_name

*call* display\_theme

*push* [color]

*push* [len]

*push* [y\_coordinate]

*push* [x\_coordinate]

*call* draw\_rect

*call* control\_flow

*call* convert\_to\_ascii\_all

*inc* [number\_of\_files]

*mov* al, [number\_of\_files]

*cmp* al, MAX\_FILES

*JE* exit

*jmp* keep\_doing\_till\_quit

exit:

*mov* ax, 4c00h

*int* 21h

END start

1. ניתן למצוא את הקוד [כאן](https://github.com/CHIPOPO64bit/rsa_asm_c). [↑](#footnote-ref-1)