**TETRIS8086  
דן מרקוביץ, 2021**

**מוסד:** תיכון הדרים 441006 **שמות המנחים:** ענבל רגב

תוכן עניינים

[תיאור השימוש 3](#_Toc74075116)

[תיאור המסכים 4](#_Toc74075117)

[מסך #1: מסך הפתיחה 4](#_Toc74075118)

[מסך #2: מסך המשחק 4](#_Toc74075119)

[מסך #3: לאחר הזזה 5](#_Toc74075120)

[מסך #4: לאחר שהטטרומינו נפל 5](#_Toc74075121)

[מסך #5-#6: שמירה (hold) ראשונית 6](#_Toc74075122)

[מסך #7-#8: שמירה (hold) שנייה 7](#_Toc74075123)

[מסך #9-#11: מחיקת שורה 8](#_Toc74075124)

[מסך #12-#13: סוף המשחק 9](#_Toc74075125)

[משתנים 10](#_Toc74075126)

[משתני הדפסת תמונה 10](#_Toc74075127)

[משתני המשחק 10](#_Toc74075128)

[משתני יצירת מספר רנדומלי 11](#_Toc74075129)

[פרוצדורות 12](#_Toc74075130)

[proc OpenBitmap 12](#_Toc74075131)

[Interupt Procedures 14](#_Toc74075132)

[proc enterGraphicMode 14](#_Toc74075133)

[proc waitForKeyPress 14](#_Toc74075134)

[proc drawPixel 15](#_Toc74075135)

[proc readPixel 15](#_Toc74075136)

[proc delay 16](#_Toc74075137)

[proc Cursor\_Location 16](#_Toc74075138)

[proc Print\_Text 17](#_Toc74075139)

[ציורים 17](#_Toc74075140)

[proc drawSquare 17](#_Toc74075141)

[פרוצדורות ציור טטרומינואים 19](#_Toc74075142)

[פרוצדורות מחיקת טטרומינואים 33](#_Toc74075143)

[לוגיקות 47](#_Toc74075144)

[proc intializeRandom 47](#_Toc74075145)

[proc randomNum 48](#_Toc74075146)

[proc generate\_last\_7\_queue 49](#_Toc74075147)

[פרוצדורות סיבוב 52](#_Toc74075148)

[פרוצדורות תזוזה 85](#_Toc74075149)

[proc generate\_piece 149](#_Toc74075150)

[proc destoy\_piece 150](#_Toc74075151)

[proc move\_down\_lines 155](#_Toc74075152)

[proc is\_game\_over 159](#_Toc74075153)

[proc draw\_queue\_thumbnails 160](#_Toc74075154)

[proc erase\_queue\_thumbnails 161](#_Toc74075155)

[proc draw\_held\_piece\_thumbnails 162](#_Toc74075156)

[proc erase\_held\_piece\_thumbnails 163](#_Toc74075157)

[proc draw\_score 164](#_Toc74075158)

[פרוצדורות הגדלת הניקוד 165](#_Toc74075159)

[proc draw\_level 166](#_Toc74075160)

[proc calculate\_level 167](#_Toc74075161)

[proc draw\_cleared\_lines 174](#_Toc74075162)

[proc inc\_cleared\_lines 175](#_Toc74075163)

# תיאור השימוש

הפרויקט שלי הוא ממימוש של משחק הטטריס.

המשחק מורכב מריבועים המסודרים ב7 הצורות אשר נקראות "הטטרומינואים החד צדדיים" הנופלות בלוח שגודלו 10 ריבועים לרוחב ו21 ריבועים לגובה.

במשחק השחקן יכול לסובב את הטטרומינואים (בערת לחיצה על המקש a או d), להזיז אותם אופקית על גבי הלוח (בעזרת לחיצה על מקשי החצים האופקיים), להאיץ את נפילת הטטרומינואים (נקרא soft drop) (בעזרת לחיצה על מקש החץ התחתון), להפיל אותם למטה (נקרא hard drop) (בעזרת מקש החץ העליון) או לשמור את הטטרומינו לשימוש מאוחר יותר (בעזרת מקש הרווח).

כאשר טטרומינו לא יכול ליפול שורה נוספת (כלומר, כאשר הוא מגיע לתחתית הלוח או כאשר ריבוע מונע ממנו ליפול) הוא נעצר במקום וטטרומינו חדש נוצר בראש הלוח

כאשר שורה של ריבועים מתמלאת, היא נמחקת וכל השורות מעליה יורדות שורה אחת למטה.

כאשר ריבוע מגיע לראש הלוח, במקום בו הטטרומינואים נוצרים, המשחק נגמר.

עבור כל מספר שורות שנמחקות המשחק עולה שלב.

השלב משפיע על המהירות בה הטטרומינואים נופלים ועל כמות הנקודות שהשחקן מקבל כאשר נמחקת שורה.

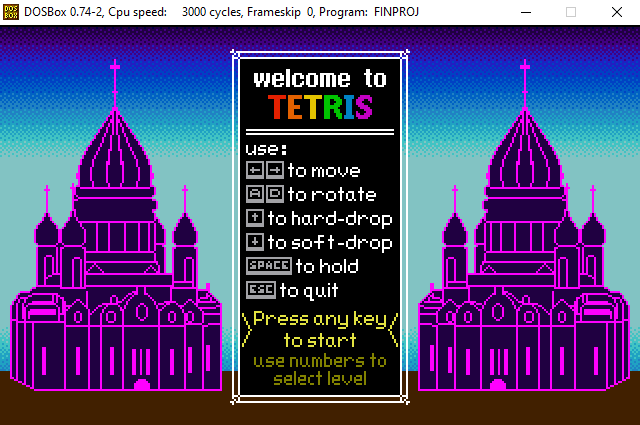
מספר שורות יכולות להימחק בבת-אחת.

השחקן יכול לקבל נקודות עבור hard drop, soft drop (נקודה עבור כל שורה שיורדת מהר יותר), או עבור מחיקת שורות (קבוע הנקודות עבור השורות שנמחקו כפול מספר מעל השלב הנוכחי).

יש לשים לב כי לכל טטרומינו יש שם, לפי האות האנגלית שדומה לצורתה, ובספר פרויקט זה אתייחס לטטרומינואים לפי שמות אלו.

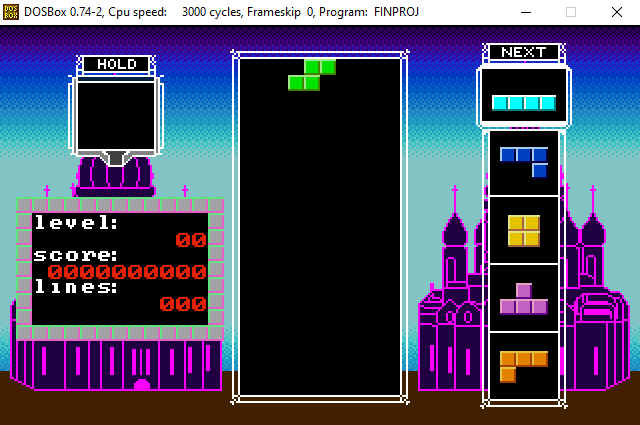
# תיאור המסכים

### מסך #1: מסך הפתיחה

**

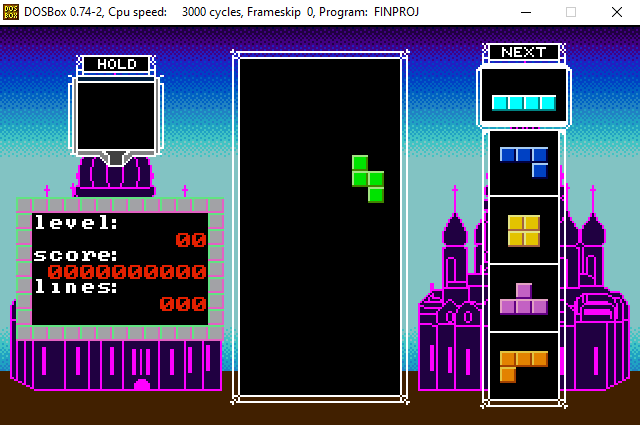
במסך זה השחקן מקבל מידע על איך לשחק ומקבל אפשרות להתחיל את המשחק בכל שלב בין 0 ל-9 (כל מקש שילחץ יוביל להתחלה בשלב 0 למעט מקשי המספרים שיובילו להתחלה בשלב בין 1 ל-9)

### מסך #2: מסך המשחק

**

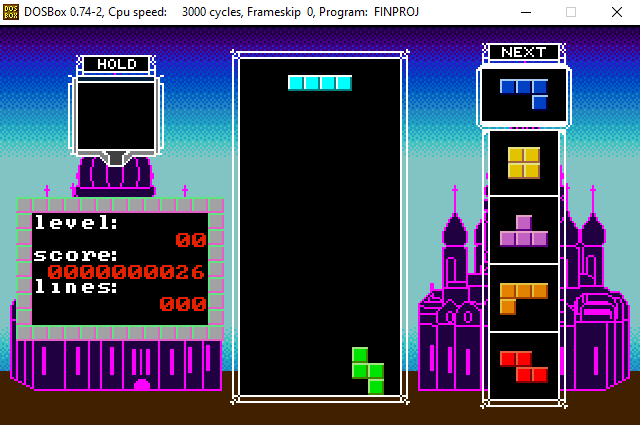
מסך זה הוא המסך שמופיע כאשר המשחק מתחיל לשחק (הטטרומינואים רנדומליים)

### מסך #3: לאחר הזזה

**

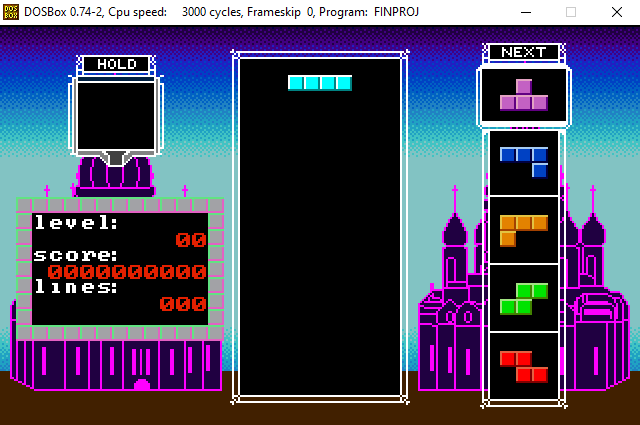
כך מסך המשחק יכול להיראות כאשר השחקן מזיז ומסובב את הטטרומינו, בנוסף לכך, הטטרומינו נפל מעט כחלק מהנפילה ה"טבעית" (שאוטומטית קורת) שלו

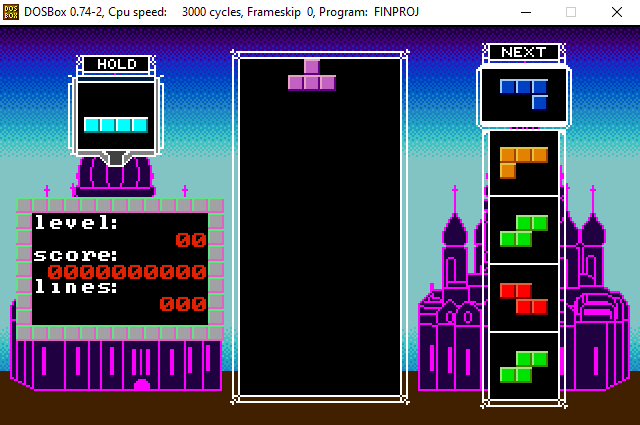
### מסך #4: לאחר שהטטרומינו נפל

**

לאחר שהטטרומינו הגיע לתחתית המסך, הוא נעצר ואחר חדש, בו כעט השחקן שולט נוצר בראש המסך

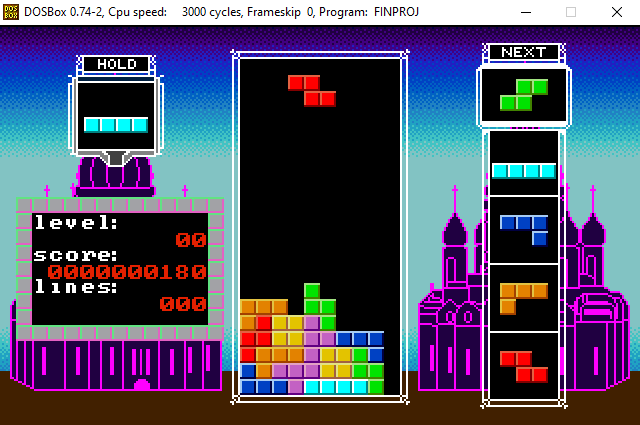
### מסך #5-#6: שמירה (hold) ראשונית

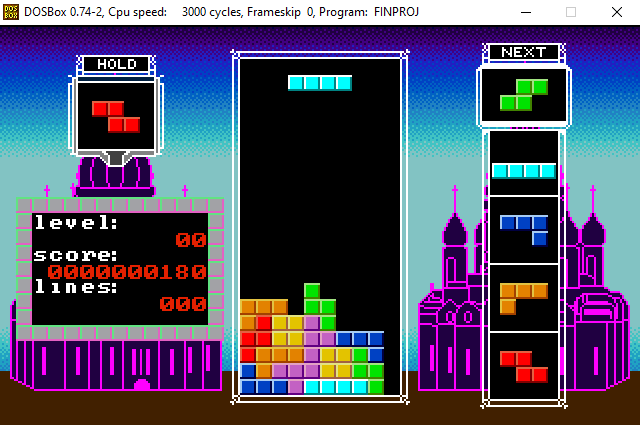
**

**

במסכים אלו ניתן לראות את תהליך הפעולה של שמירה (hold), במסך הראשון ניתן לראות את טטרומינו ה-I אשר אותו השחקן רוצה לשמור למצב מאוחר יותר. כעט אין טטרומינו בתא השמירה לכן במסך השני, ניתן לראות כי טטרומינו ה-I נכנס לתא השמירה, מכיוון שאין שום טטרומינו על המסך, הטטרומינו הבא בתור (טטרומינו ה-T במקרה זה) נוצר בראש המסך

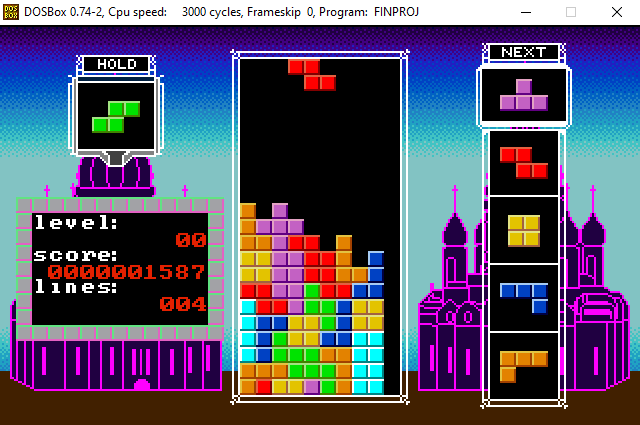
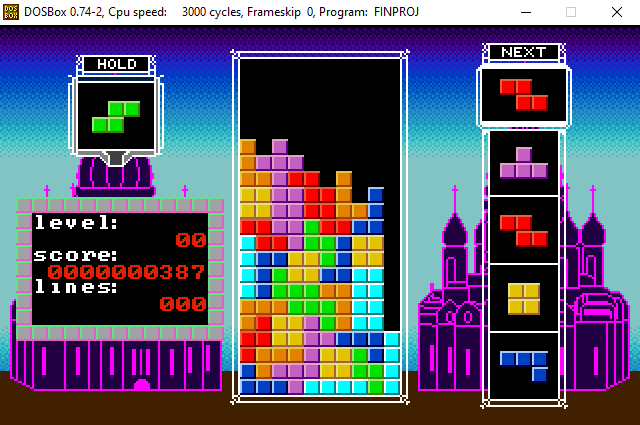
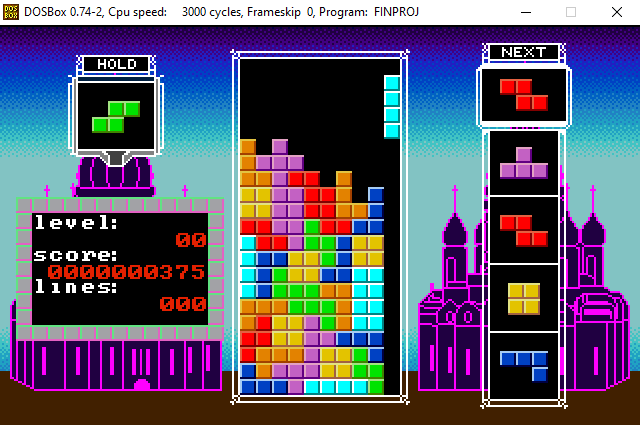
### מסך #7-#8: שמירה (hold) שנייה





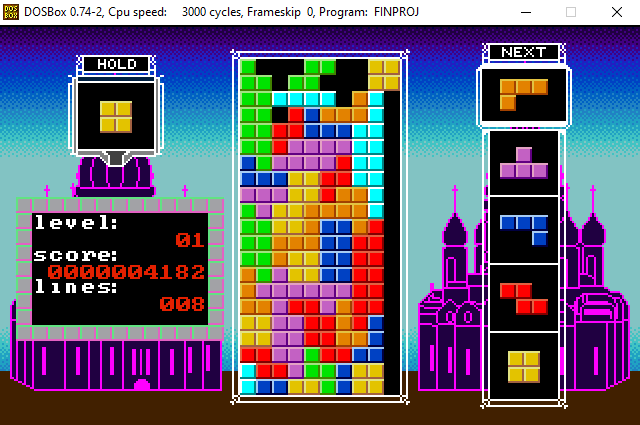
לאחר שהשחקן כבר שמר טטרומינו, והוא רוצה לשמור עוד אחד במקומו (מסך 7), הוא יכול ללחוץ רווח עוד פעם, והטטרומינו בתא השמירה מתחלף עם זה שעל המסך הראשי (מסך 8).

### מסך #9-#11: מחיקת שורה



במסכים 9, 10 ניתן לראות את השחקן מוריד את הטטרומינו I עד לתחתית הלוח, ולאחר מכן, במסך 11, ניתן לראות כי השורות שהתמלאו נמחקו. בנוסף לכך, הניקוד עלה בצורה דראסטית וסטטוס השורות עלה ב-4 (מספר השורות שנמחקו).

### מסך #12-#13: סוף המשחק



במסך 12, ניתן לראות כי טטרומינו ה-S אשר הונח כעט חוסם את האזור בו טטרומינואים "רוצים" להיווצר, לכן המשחק לא יכול להמשיך והשחקן מגיע למסך הסוף (מסך 13) בו הוא יכול לראות את באיזה שלב הפסיד, מה היה הניקוד שלו, וכמה שורות הוא ניקה. למסך 13 ניתן להגיע בעזרת לחיצה על כפתור ה-escape בכל רגע במהלך המשחק. כאשר לוחצים במסך 13 על כל מקש, התוכנית נסגרת.

# משתנים

### משתני הדפסת תמונה

|  |  |  |  |
| --- | --- | --- | --- |
| ערכים התחלתיים | הסבר | גודל | שם משתנה |
| אין | המיקום בזיכרון של שם התמונה שהתוכנה רוצה להדפיס | dw | filename |
| 'pic2019.bmp',0 | שם התמונה הראשונה (הרקע של המשחק) | db | filename1 |
| 'pic2020.bmp',0 | שם התמונה השנייה (מסך הפתיחה) | db | filename2 |
| 'pic2021.bmp',0 | שם התמונה השלישית (מסך הסוף) | db | filename3 |
| אין | מצביע למקום הקובץ שנפתח בזיכרון | dw | filehandle |
| 54 dup (0) | 54 הבייטים הראשונים של הקובץ | db | Header |
| 256\*4 dup (0) | פלטת הצבעים של קבצי הBMP | db | Palette |
| 320 dup (0) | השורה הנוכחית בקריאת קובץ הBMP | db | ScrLine |
| 'Error', 13, 10, '$' | הודעת שגיאה אשר תודפס כאשר התוכנה לא מצליחה לקרוא את הקובץ | db | ErrorMsg |

### משתני המשחק

|  |  |  |  |
| --- | --- | --- | --- |
| ערכים התחלתיים | הסבר | גודל | שם משתנה |
| אין | קואורדינטת ה-X כאשר רוצים לקרוא/להדפיס פיקסל על המסך | dw | x\_coordinate |
| אין | קואורדינטת ה-Y כאשר רוצים לקרוא/להדפיס פיקסל על המסך | dw | y\_coordinate |
| אין | צבע הפיקסל שהתוכנה רוצה להדפיס | dw | colour |
| אין | צבע הפיקסל כאשר קוראים פיקסל מהמסך | db | pixelColour |
| אין | מקש שנלחץ | db | pressedKey |
| 8 | גודל ריבוע | db | square\_size |
| אין | צבע ראשי (בשימוש בפעולת ציור הריבועים) | dw | main\_colour |
| אין | הצבע הכהה (בשימוש בפעולת ציור הריבועים) | dw | border\_colour |
| אין | הצבע הבהיר (בשימוש בפעולת ציור הריבועים) | dw | light\_colour |
| אין | סוג הטטרומינו בו השחקן שולט | dw | current\_piece |
| 1 | מספר הסיבוב הנוכחי של הטטרומינו בו השחקן שולט | dw | current\_piece\_rotation |
| 0ffffh | המהירות בה הטטרומינואים נופלים | dw | move\_down\_speed |
| 0ffffh | המהירות בה הטטרומינואים נופלים לפי השלב | dw | default\_speed |
| 0 | משתנה אשר אומר האם פעולת נפילת הטטרומינו הנכשלה (0=לא, 1=כן) | db | move\_down\_failed |
| 0 | משתנה אשר אומר האם מקש החץ העליון נלחץ בתור הזה (0=לא, 1=כן) | db | up\_key\_pressed |
| 0 | מספר השורה הנוכחי (משמש למערכת שמנקה שורות) | dw | line |
| 0 | משתנה אשר אומר האם המשחק נגמר (0=לא, 1=כן) | db | game\_over |
| 100 | הטטרומינו שנשמר | dw | held\_piece |
| 0 | משתנה אשר אומר האם נשמרה כבר חתיכה בתור זה (0=לא, 1=כן) | dw | held\_this\_turn |
| 14 dupe (?) | רשימת הטטרומינואים שיגיעו אחרי האחד הזה~ | dw | queue |
| 14 | משתנה עזר לפעולת טעינת התור, אומר מה כרגע המקום הכי נמוך בתור שאינו פנוי | db | min\_queue\_last\_7 |
| 0 | כמה תורות עברו מאז שהתור התמלא | db | queue\_iteration |
| 10 dup(0), "$" | הניקוד של השחקן | db | score |
| 0 | כמה שורות נוקו בתור הזה | db | lines\_cleard\_this\_turn |
| 2 dup(0), "$" | השלב הנוכחי בצורת מערך (כדי שיהיה נוח להדפיס) | db | Level |
| 0 | כמה שורות השחקן ניקה | dw | lines\_cleared |
| 3 dup(0), "$" | כמה שורות השחקן ניקה בצורה שנוח להדפיס | db | lines\_cleared\_printable |

### משתני יצירת מספר רנדומלי

|  |  |  |  |
| --- | --- | --- | --- |
| ערכים התחלתיים | הסבר | גודל | שם משתנה |
| 6075 | קבועים בנוסחה של הראנדום | dw | modulus |
| 106 | dw | multiplier |
| 1283 | dw | increment |
| אין | הזרע של הראנדום | dw | seed |
| 7 | מספר מעל המספר המקסימלי שאליו יכול להגיע המספר הרנדומלי | dw | top\_limit |
| ? | המספר הרנדומלי שהתקבל | dw | rand\_num |

# פרוצדורות

### proc OpenBitmap

**תיאור הפרוצדורה**  
פרוצדורה זו קוראת קובץ פותחת קובץ BMP ומדפיסה אותו על המסך

**משתנים**filename – שם הקובץ שצריך לפתוח  
filehandle – המקום בזכרון בו נשמר הקובץ אחרי שהוא נפתח  
ErrorMsg – הודעת שגיאה במקרה שהקובץ לא נפתח  
Header – הכותרת של הקובץ  
Palette – הפלטה של הצבעים בBMP  
ScrLine – הBMP מועתק למסך שורה אחר שורה

**קוד הפרוצדורה**

**proc** OpenBitmap

    ; Open file

**mov** **ah**, 3Dh

**xor** **al**, **al**

**mov** **dx**, [filename]

**int** 21h

**jc** openerror

**mov** [filehandle], **ax**

**jmp** readheader

    openerror:

**mov** **dx**, **offset** ErrorMsg

**mov** **ah**, 9h

**int** 21h

**ret**

    readHeader:

    ; Read BMP file header, 54 bytes

**mov** **ah**,3fh

**mov** **bx**, [filehandle]

**mov** **cx**,54

**mov** **dx**,**offset** Header

**int** 21h

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

**mov** **ah**,3fh

**mov** **cx**,400h

**mov** **dx**,**offset** Palette

**int** 21h

    ; 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

**mov** **si**,**offset** Palette

**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 video 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.)

**loop** PalLoop

    ; BMP graphics are saved upside-down.

    ; Read the graphic line by line (200 lines in VGA format),

    ; displaying the lines from bottom to top.

**mov** **ax**, 0A000h

**mov** **es**, **ax**

**mov** **cx**,200

    PrintBMPLoop:

**push** **cx**

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

**mov** **di**,**cx**

**shl** **cx**,6

**shl** **di**,8

**add** **di**,**cx**

    ; Read one line

**mov** **ah**,3fh

**mov** **cx**,320

**mov** **dx**,**offset** ScrLine

**int** 21h

    ; Copy one line into video memory

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

**mov** **cx**,320

**mov** **si**,**offset** ScrLine

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

    ;rep movsb is same as the following code:

    ;mov es:di, ds:si

    ;inc si

    ;inc di

    ;dec cx

    ;loop until cx=0

**pop** **cx**

**loop** PrintBMPLoop

**ret**

**endp** OpenBitmap

## Interupt Procedures

### proc enterGraphicMode

**תיאור הפרוצדורה**פרוצדורה זו מכניסה את המחשב למצב גרפי, משמשת גם למחיקת המסך

**קוד הפרוצדורה**

**proc** enterGraphicMode

**push** **ax**

    ; graphic mode

**mov** **ax**, 13h

**int** 10h

**pop** **ax**

**ret**

**endp** entergraphicmode

### proc waitForKeyPress

**תיאור הפרוצדורה**פרוצדורה זו מחכה למקש ושומרת איזה מקש נלחץ

**משתנים**pressedKey – המקש שנלחץ

**קוד הפרוצדורה**

**proc** waitForKeyPress

**push** **ax**

    ; wait for key

**mov** **ah**, 0h

**int** 16h

**mov** [pressedKey], **al**

**pop** **ax**

**ret**

**endp** waitForKeyPress

### proc drawPixel

**תיאור הפרוצדורה**פרוצדורה זו מדפיסה פיקסל

**משתנים**  
x\_coordinate – קואורדינטת ה-x בה יודפס הפיקסל  
y\_coordinate – קואורדינטת ה-y בה יודפס הפיקסל  
[bp+4] – צבע הפיקסל

**קוד הפרוצדורה**

**proc** drawPixel

**push** **bp**

**mov** **bp**,**sp**

**pusha**

    ; print pixel interrupt

**xor** **bh**, **bh** ; bh = 0

**mov** **cx**, [x\_coordinate] ; x coord

**mov** **dx**, [y\_coordinate] ; y coord

**mov** **ax**, [**bp+**4] ; colour

**mov** **ah**, 0ch

**int** 10h

**popa**

**pop** **bp**

**ret** 2

**endp** drawPixel

### proc readPixel

**תיאור הפרוצדורה**  
פרוצדורה זו בודקת מה הצבע של הפיקסל במקום מסוים

**משתנים**x\_coordinate – קואורדינטת ה-x בה יבדק הפיקסל  
y\_coordinate – קואורדינטת ה-y בה יבדק הפיקסל  
pixelColour – הצבע של הפיקסל

**קוד הפרוצדורה**

**proc** readPixel

**pusha**

**mov** **cx**, [x\_coordinate] ; x coord

**mov** **dx**, [y\_coordinate] ; y coord

**mov** **ah**, 0Dh ; read colour interrupt

**int** 10h

**mov** [pixelcolour], **al**

**popa**

**ret**

**endp** readPixel

### proc delay

**תיאור הפרוצדורה**פרוצדורה זו עוצרת את המחשב לכמות זמן מוגדרת

**משתנים**move\_down\_speed – כמות הזמן שתיעצר במיקרו שניות

**קוד הפרוצדורה**

**proc** delay

**pusha**

**mov** **cx**, 0h   ; High Word

**mov** **dx**, [move\_down\_speed]   ;Low Word

**mov** **al**, 0

**mov** **ah**, 86h  ; Wait function

**int** 15h

**popa**

**ret**

**endp** delay

### proc Cursor\_Location

**תיאור הפרוצדורה**פרוצדורה זו קובעת את מיקום המצביע של הטקסט

**משתנים**  
[bp+6] – קואורדינטת ה-x של הסמן  
[bp+4] – קואורדינטת ה-y של הסמן

**קוד הפרוצדורה**

local\_x equ [**bp+**6]

local\_y equ [**bp+**4]

**proc** Cursor\_Location ;Place the cursor on the screen by bp

**push** **bp**

**mov** **bp**,**sp**

**pusha**

    ; set cursor location

**mov** **bh**, 0

**mov** **dl**, local\_x ; in column/x

**mov** **dh**, local\_y ; in row/y

**mov** **ah**, 2

**int** 10h

**popa**

**pop** **bp**

**ret** 4

**endp** Cursor\_Location

### proc Print\_Text

**תיאור הפרוצדורה**  
מדפיסה טקסט במיקום הסמן

**משתנים**dx – הטקסט להדפסה

**קוד הפרוצדורה**

**proc** Print\_Text ;print text in dx

**pusha**

**mov** **ah**, 9h

**int** 21h

**popa**

**ret**

**endp** Print\_Text

## ציורים

### proc drawSquare

**תיאור הפרוצדורה**פרוצדורה זו מציירת ריבוע בעזרת פיקסלים

**משתנים**x\_coordinate – קואורדינטת ה-x בה יודפס הפיקסל  
y\_coordinate – קואורדינטת ה-y בה יודפס הפיקסל  
[bp+4] – צבע הפיקסל  
main\_colour – הצבע הראשי של הריבוע  
border\_colour – הצבע של הצל של הריבוע  
light\_colour – הצבע של האור על הריבוע

**קוד הפרוצדורה**

**proc** drawSquare

**push** **cx**

        ; draw a basic square using the given colours

        ; outer square

**push** [y\_coordinate]

**mov** **cx**, [square\_size] ; set column loop counter

        drawSquare\_column:

**push** **cx** ; push to not lose big loop counter

**push** [x\_coordinate] ; in order to reset the x\_coord every row

**mov** **cx**, [square\_size] ; set row loop counter

            drawSquare\_row:

**push** [main\_colour]

**call** drawpixel ; draw pixel

**inc** [x\_coordinate]

**loop** drawsquare\_row ; loop for the whole row

**pop** [x\_coordinate] ; reset x\_coord

**pop** **cx** ; get big loop counter back

**inc** [y\_coordinate] ; next row

**loop** drawsquare\_column

**pop** [y\_coordinate] ; reset y\_coord

        ;border

**push** [x\_coordinate]

**push** [y\_coordinate]

**mov** **cx**, [square\_size]

        drawSquare\_border\_top:

**push** [light\_colour]

**call** drawpixel ; draw pixel

**inc** [x\_coordinate]

**loop** drawsquare\_border\_top ; loop for the whole row

**dec** [x\_coordinate]

**inc** [y\_coordinate]

**mov** **cx**, [square\_size]

**dec** **cx**

        drawSquare\_border\_right:

**push** [border\_colour]

**call** drawpixel ; draw pixel

**inc** [y\_coordinate]

**loop** drawsquare\_border\_right ; loop for the whole column

**dec** [y\_coordinate]

**mov** **cx**, [square\_size]

        drawSquare\_border\_bottom:

**push** [border\_colour]

**call** drawpixel ; draw pixel

**dec** [x\_coordinate]

**loop** drawsquare\_border\_bottom ; loop for the whole row

**inc** [x\_coordinate]

**dec** [y\_coordinate]

**mov** **cx**, [square\_size]

**dec** **cx**

        drawSquare\_border\_left:

**push** [light\_colour]

**call** drawpixel ; draw pixel

**dec** [y\_coordinate]

**loop** drawsquare\_border\_left ; loop for the whole column

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**pop** **cx**

**ret**

**endp** drawSquare

### פרוצדורות ציור טטרומינואים

**תיאור הפרוצדורות**פרוצדורות אלו מציירות את הטטרומינואים השונים

**משתנים**main\_colour – הצבע העיקרי של הטטרומינו  
light\_colour – הצבע הבהיר של הטטרומינו  
border\_colour - הצבע הכהה של הטטרומינו

**קוד הפרוצדורות**

**proc** drawTPiece\_1

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

        ;  O

        ; OOO

**mov** [light\_colour], 0efh

**mov** [main\_colour], 0deh

**mov** [border\_colour], 83h

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [x\_coordinate], **ax** ; top square position

**call** drawsquare ; draw top square

**sub** [x\_coordinate], **ax** ; bottom squares position

**add** [y\_coordinate], **ax**

**mov** **cx**, 3 ; draw bottom 3 pieces

        drawTPiece\_1\_bottomLoop:

**call** drawsquare

**add** [x\_coordinate], **ax** ; move to next

**loop** drawTPiece\_1\_bottomLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawTPiece\_1

**proc** drawTPiece\_2

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

        ;  O

        ; OO

        ;  O

**mov** [light\_colour], 0efh

**mov** [main\_colour], 0deh

**mov** [border\_colour], 83h

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [y\_coordinate], **ax** ; left square position

**call** drawsquare ; draw left square

**sub** [y\_coordinate], **ax** ; right squares position

**add** [x\_coordinate], **ax**

**mov** **cx**, 3 ; draw middle 3 pieces

        drawTPiece\_2\_middleLoop:

**call** drawsquare

**add** [y\_coordinate], **ax** ; move to next

**loop** drawTPiece\_2\_middleLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawTPiece\_2

**proc** drawTPiece\_3

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

        ; OOO

        ;  O

**mov** [light\_colour], 0efh

**mov** [main\_colour], 0deh

**mov** [border\_colour], 83h

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [x\_coordinate], **ax** ; bottom square position

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** drawsquare ; draw bottom square

**sub** [x\_coordinate], **ax** ; top squares position

**sub** [y\_coordinate], **ax**

**mov** **cx**, 3 ; draw top 3 pieces

        drawTPiece\_3\_topLoop:

**call** drawsquare

**add** [x\_coordinate], **ax** ; move to next

**loop** drawTPiece\_3\_topLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawTPiece\_3

**proc** drawTPiece\_4

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

        ;  O

        ;  OO

        ;  O

**mov** [light\_colour], 0efh

**mov** [main\_colour], 0deh

**mov** [border\_colour], 83h

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [y\_coordinate], **ax** ; right square position

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** drawsquare ; draw right square

**sub** [y\_coordinate], **ax** ; middle squares position

**sub** [x\_coordinate], **ax**

**mov** **cx**, 3 ; draw middle 3 pieces

        drawTPiece\_4\_middleLoop:

**call** drawsquare

**add** [y\_coordinate], **ax** ; move to next

**loop** drawTPiece\_4\_middleLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawTPiece\_4

**proc** drawOPiece

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **ax**

    ; OO

    ; OO

**mov** **ax**, [square\_size]

    ; o-piece colours

**mov** [main\_colour], 37h ; orangish yellow

**mov** [light\_colour], 0bfh ; light yellow

**mov** [border\_colour], 5dh ; brown

**add** [x\_coordinate], **ax**

**call** drawsquare ; top left

**add** [x\_coordinate], **ax**

**call** drawsquare ; top right

**add** [y\_coordinate], **ax**

**call** drawsquare ; bottom right

**sub** [x\_coordinate], **ax**

**call** drawsquare ; bottom left

**pop** **ax**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawopiece

**proc** drawJPiece\_1

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ; OOO

        ;   O

**mov** [light\_colour], 9h ; blues

**mov** [main\_colour], 0d0h

**mov** [border\_colour], 40h

**mov** **cx**, 3 ; draw top 3 pieces

        drawJPiece\_1\_topLoop:

**call** drawsquare

**add** [x\_coordinate], **ax** ; move to next

**loop** drawJPiece\_1\_topLoop

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax** ; bottom square position

**call** drawsquare ; draw bottom square

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawJPiece\_1

**proc** drawJPiece\_2

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ; OO

        ; O

        ; O

**mov** [light\_colour], 9h ; blues

**mov** [main\_colour], 0d0h

**mov** [border\_colour], 40h

**add** [x\_coordinate], **ax** ; right square position

**call** drawsquare

**sub** [x\_coordinate], **ax** ; left squares position

**mov** **cx**, 3 ; draw left 3 pieces

        drawJPiece\_2\_leftLoop:

**call** drawsquare

**add** [y\_coordinate], **ax** ; move to next

**loop** drawJPiece\_2\_leftLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawJPiece\_2

**proc** drawJPiece\_3

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ;

        ; O

        ; OOO

**mov** [light\_colour], 9h ; blues

**mov** [main\_colour], 0d0h

**mov** [border\_colour], 40h

**add** [y\_coordinate], **ax** ; top square position

**call** drawsquare ; draw top square

**add** [y\_coordinate], **ax** ; top square position

**mov** **cx**, 3 ; draw bottom 3 pieces

        drawJPiece\_3\_bottomLoop:

**call** drawsquare

**add** [x\_coordinate], **ax** ; move to next

**loop** drawJPiece\_3\_bottomLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawJPiece\_3

**proc** drawJPiece\_4

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ;   O

        ;   O

        ;  OO

**mov** [light\_colour], 9h ; blues

**mov** [main\_colour], 0d0h

**mov** [border\_colour], 40h

**add** [x\_coordinate], **ax** ; right squares position

**add** [x\_coordinate], **ax**

**mov** **cx**, 3 ; draw right 3 pieces

        drawJPiece\_4\_rightLoop:

**call** drawsquare

**add** [y\_coordinate], **ax** ; move to next

**loop** drawJPiece\_4\_rightLoop

**sub** [y\_coordinate], **ax** ; left square position

**sub** [x\_coordinate], **ax**

**call** drawsquare

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawJPiece\_4

**proc** drawLPiece\_1

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ; OOO

        ; O

**mov** [light\_colour], 77h ; orange

**mov** [main\_colour], 27h

**mov** [border\_colour], 15h

**add** [y\_coordinate], **ax**

**call** drawsquare ; draw bottom square

**sub** [y\_coordinate], **ax**

**mov** **cx**, 3 ; draw top 3 pieces

        drawLPiece\_1\_topLoop:

**call** drawsquare

**add** [x\_coordinate], **ax** ; move to next

**loop** drawLPiece\_1\_topLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawLPiece\_1

**proc** drawLPiece\_2

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ; O

        ; O

        ; OO

**mov** [light\_colour], 77h ; orange

**mov** [main\_colour], 27h

**mov** [border\_colour], 15h

**mov** **cx**, 3 ; draw left 3 pieces

        drawLPiece\_2\_leftLoop:

**call** drawsquare

**add** [y\_coordinate], **ax** ; move to next

**loop** drawLPiece\_2\_leftLoop

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** drawsquare ; draw bottom square

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawLPiece\_2

**proc** drawLPiece\_3

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ;

        ;   O

        ; OOO

**mov** [light\_colour], 77h ; orange

**mov** [main\_colour], 27h

**mov** [border\_colour], 15h

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**mov** **cx**, 3 ; draw bottom 3 pieces

        drawLPiece\_3\_bottomLoop:

**call** drawsquare

**add** [x\_coordinate], **ax** ; move to next

**loop** drawLPiece\_3\_bottomLoop

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** drawsquare ; draw bottom square

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawLPiece\_3

**proc** drawLPiece\_4

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ;  OO

        ;   O

        ;   O

**mov** [light\_colour], 77h ; orange

**mov** [main\_colour], 27h

**mov** [border\_colour], 15h

**add** [x\_coordinate], **ax**

**call** drawsquare ; draw bottom square

**add** [x\_coordinate], **ax**

**mov** **cx**, 3 ; draw right 4 pieces

        drawLPiece\_4\_rightLoop:

**call** drawsquare

**add** [y\_coordinate], **ax** ; move to next

**loop** drawLPiece\_4\_rightLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawLPiece\_4

**proc** drawIPiece\_1

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

        ;

        ; OOOO

**mov** [light\_colour], 0ffh ; cyan

**mov** [main\_colour], 0feh

**mov** [border\_colour], 6h

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [y\_coordinate], **ax**

**mov** **cx**, 4 ; draw line

        drawIPiece\_1\_bottomLoop:

**call** drawsquare

**add** [x\_coordinate], **ax** ; move to next

**loop** drawIPiece\_1\_bottomLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawIPiece\_1

**proc** drawIPiece\_2

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

        ; O

        ; O

        ; O

        ; O

**mov** [light\_colour], 0ffh ; cyan

**mov** [main\_colour], 0feh

**mov** [border\_colour], 6h

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [x\_coordinate], **ax**

**mov** **cx**, 4 ; draw line

        drawIPiece\_2\_leftLoop:

**call** drawsquare

**add** [y\_coordinate], **ax** ; move to next

**loop** drawIPiece\_2\_leftLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawIPiece\_2

**proc** drawSPiece\_1

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **ax**

        ;  OO

        ; OO

**mov** [light\_colour], 0bdh ; greens

**mov** [main\_colour], 38h

**mov** [border\_colour], 22h

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [y\_coordinate], **ax**

**call** drawsquare

**add** [x\_coordinate], **ax**

**call** drawsquare

**sub** [y\_coordinate], **ax**

**call** drawsquare

**add** [x\_coordinate], **ax**

**call** drawsquare

**pop** **ax**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawSPiece\_1

**proc** drawSPiece\_2

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **ax**

        ; O

        ; OO

        ;  O

**mov** [light\_colour], 0bdh ; greens

**mov** [main\_colour], 38h

**mov** [border\_colour], 22h

**mov** **ax**, [square\_size] ; mov square size to a register

**call** drawsquare

**add** [y\_coordinate], **ax**

**call** drawsquare

**add** [x\_coordinate], **ax**

**call** drawsquare

**add** [y\_coordinate], **ax**

**call** drawsquare

**pop** **ax**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawSPiece\_2

**proc** drawZPiece\_1

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **ax**

        ; OO

        ;  OO

**mov** [light\_colour], 5fh ; reds

**mov** [main\_colour], 0f9h

**mov** [border\_colour], 01h

**mov** **ax**, [square\_size] ; mov square size to a register

**call** drawsquare

**add** [x\_coordinate], **ax**

**call** drawsquare

**add** [y\_coordinate], **ax**

**call** drawsquare

**add** [x\_coordinate], **ax**

**call** drawsquare

**pop** **ax**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawZPiece\_1

**proc** drawZPiece\_2

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **ax**

        ;  O

        ; OO

        ; O

**mov** [light\_colour], 5fh ; reds

**mov** [main\_colour], 0f9h

**mov** [border\_colour], 01h

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [x\_coordinate], **ax**

**call** drawsquare

**add** [y\_coordinate], **ax**

**call** drawsquare

**sub** [x\_coordinate], **ax**

**call** drawsquare

**add** [y\_coordinate], **ax**

**call** drawsquare

**pop** **ax**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** drawZPiece\_2

### פרוצדורות מחיקת טטרומינואים

**תיאור הפרוצדורות**פרוצדורות אלו מציירות את הטטרומינואים השונים

**קוד הפרוצדורות  
proc** blackTPiece\_1

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

        ;  O

        ; OOO

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**mov** [light\_colour], 0

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [x\_coordinate], **ax** ; top square position

**call** drawsquare ; draw top square

**sub** [x\_coordinate], **ax** ; bottom squares position

**add** [y\_coordinate], **ax**

**mov** **cx**, 3 ; draw bottom 3 pieces

        blackTPiece\_1\_bottomLoop:

**call** drawsquare

**add** [x\_coordinate], **ax** ; move to next

**loop** blackTPiece\_1\_bottomLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackTPiece\_1

**proc** blackTPiece\_2

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

        ;  O

        ; OO

        ;  O

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**mov** [light\_colour], 0

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [y\_coordinate], **ax** ; left square position

**call** drawsquare ; black left square

**sub** [y\_coordinate], **ax** ; right squares position

**add** [x\_coordinate], **ax**

**mov** **cx**, 3 ; black middle 3 pieces

        blackTPiece\_2\_middleLoop:

**call** drawsquare

**add** [y\_coordinate], **ax** ; move to next

**loop** blackTPiece\_2\_middleLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackTPiece\_2

**proc** blackTPiece\_3

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

        ; OOO

        ;  O

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**mov** [light\_colour], 0

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [x\_coordinate], **ax** ; bottom square position

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** drawsquare ; black bottom square

**sub** [x\_coordinate], **ax** ; top squares position

**sub** [y\_coordinate], **ax**

**mov** **cx**, 3 ; black top 3 pieces

        blackTPiece\_3\_topLoop:

**call** drawsquare

**add** [x\_coordinate], **ax** ; move to next

**loop** blackTPiece\_3\_topLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackTPiece\_3

**proc** blackTPiece\_4

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

        ;  O

        ;  OO

        ;  O

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**mov** [light\_colour], 0

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [y\_coordinate], **ax** ; right square position

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** drawsquare ; black right square

**sub** [y\_coordinate], **ax** ; middle squares position

**sub** [x\_coordinate], **ax**

**mov** **cx**, 3 ; black middle 3 pieces

        blackTPiece\_4\_middleLoop:

**call** drawsquare

**add** [y\_coordinate], **ax** ; move to next

**loop** blackTPiece\_4\_middleLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackTPiece\_4

**proc** blackOPiece

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **ax**

    ; OO

    ; OO

**mov** **ax**, [square\_size]

    ; o-piece colours

**mov** [main\_colour], 0 ; black

**mov** [light\_colour], 0

**mov** [border\_colour], 0

**add** [x\_coordinate], **ax**

**call** drawsquare ; top left

**add** [x\_coordinate], **ax**

**call** drawsquare ; top right

**add** [y\_coordinate], **ax**

**call** drawsquare ; bottom right

**sub** [x\_coordinate], **ax**

**call** drawsquare ; bottom left

**pop** **ax**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackopiece

**proc** blackJPiece\_1

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ; OOO

        ;   O

**mov** [light\_colour], 0 ; blacks

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**mov** **cx**, 3 ; black top 3 pieces

        blackJPiece\_1\_topLoop:

**call** drawsquare

**add** [x\_coordinate], **ax** ; move to next

**loop** blackJPiece\_1\_topLoop

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax** ; bottom square position

**call** drawsquare ; black bottom square

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackJPiece\_1

**proc** blackJPiece\_2

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ; OO

        ; O

        ; O

**mov** [light\_colour], 0 ; blacks

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**add** [x\_coordinate], **ax** ; right square position

**call** drawsquare

**sub** [x\_coordinate], **ax** ; left squares position

**mov** **cx**, 3 ; black left 3 pieces

        blackJPiece\_2\_leftLoop:

**call** drawsquare

**add** [y\_coordinate], **ax** ; move to next

**loop** blackJPiece\_2\_leftLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackJPiece\_2

**proc** blackJPiece\_3

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ;

        ; O

        ; OOO

**mov** [light\_colour], 0 ; blacks

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**add** [y\_coordinate], **ax** ; top square position

**call** drawsquare ; black top square

**add** [y\_coordinate], **ax** ; top square position

**mov** **cx**, 3 ; black bottom 3 pieces

        blackJPiece\_3\_bottomLoop:

**call** drawsquare

**add** [x\_coordinate], **ax** ; move to next

**loop** blackJPiece\_3\_bottomLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackJPiece\_3

**proc** blackJPiece\_4

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ;   O

        ;   O

        ;  OO

**mov** [light\_colour], 0 ; blacks

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**add** [x\_coordinate], **ax** ; right squares position

**add** [x\_coordinate], **ax**

**mov** **cx**, 3 ; black right 3 pieces

        blackJPiece\_4\_rightLoop:

**call** drawsquare

**add** [y\_coordinate], **ax** ; move to next

**loop** blackJPiece\_4\_rightLoop

**sub** [y\_coordinate], **ax** ; left square position

**sub** [x\_coordinate], **ax**

**call** drawsquare

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackJPiece\_4

**proc** blackLPiece\_1

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ; OOO

        ; O

**mov** [light\_colour], 0 ; black

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**add** [y\_coordinate], **ax**

**call** drawsquare ; black bottom square

**sub** [y\_coordinate], **ax**

**mov** **cx**, 3 ; black top 3 pieces

        blackLPiece\_1\_topLoop:

**call** drawsquare

**add** [x\_coordinate], **ax** ; move to next

**loop** blackLPiece\_1\_topLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackLPiece\_1

**proc** blackLPiece\_2

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ; O

        ; O

        ; OO

**mov** [light\_colour], 0 ; black

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**mov** **cx**, 3 ; black left 3 pieces

        blackLPiece\_2\_leftLoop:

**call** drawsquare

**add** [y\_coordinate], **ax** ; move to next

**loop** blackLPiece\_2\_leftLoop

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** drawsquare ; black bottom square

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackLPiece\_2

**proc** blackLPiece\_3

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ;

        ;   O

        ; OOO

**mov** [light\_colour], 0 ; black

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**mov** **cx**, 3 ; black bottom 3 pieces

        blackLPiece\_3\_bottomLoop:

**call** drawsquare

**add** [x\_coordinate], **ax** ; move to next

**loop** blackLPiece\_3\_bottomLoop

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** drawsquare ; black bottom square

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackLPiece\_3

**proc** blackLPiece\_4

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

**mov** **ax**, [square\_size] ; mov square size to a register

        ;  OO

        ;   O

        ;   O

**mov** [light\_colour], 0 ; black

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**add** [x\_coordinate], **ax**

**call** drawsquare ; black bottom square

**add** [x\_coordinate], **ax**

**mov** **cx**, 3 ; draw right 4 pieces

        blackLPiece\_4\_rightLoop:

**call** drawsquare

**add** [y\_coordinate], **ax** ; move to next

**loop** blackLPiece\_4\_rightLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackLPiece\_4

**proc** blackIPiece\_1

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

        ;

        ; OOOO

**mov** [light\_colour], 0 ; black

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [y\_coordinate], **ax**

**mov** **cx**, 4 ; black line

        blackIPiece\_1\_bottomLoop:

**call** drawsquare

**add** [x\_coordinate], **ax** ; move to next

**loop** blackIPiece\_1\_bottomLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackIPiece\_1

**proc** blackIPiece\_2

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **cx**

**push** **ax**

        ; O

        ; O

        ; O

        ; O

**mov** [light\_colour], 0 ; black

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [x\_coordinate], **ax**

**mov** **cx**, 4 ; black line

        blackIPiece\_2\_leftLoop:

**call** drawsquare

**add** [y\_coordinate], **ax** ; move to next

**loop** blackIPiece\_2\_leftLoop

**pop** **ax**

**pop** **cx**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackIPiece\_2

**proc** blackSPiece\_1

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **ax**

        ;  OO

        ; OO

**mov** [light\_colour], 0h ; black

**mov** [main\_colour], 0h

**mov** [border\_colour], 0h

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [y\_coordinate], **ax**

**call** drawsquare

**add** [x\_coordinate], **ax**

**call** drawsquare

**sub** [y\_coordinate], **ax**

**call** drawsquare

**add** [x\_coordinate], **ax**

**call** drawsquare

**pop** **ax**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackSPiece\_1

**proc** blackSPiece\_2

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **ax**

        ; O

        ; OO

        ;  O

**mov** [light\_colour], 0h ; black

**mov** [main\_colour], 0h

**mov** [border\_colour], 0h

**mov** **ax**, [square\_size] ; mov square size to a register

**call** drawsquare

**add** [y\_coordinate], **ax**

**call** drawsquare

**add** [x\_coordinate], **ax**

**call** drawsquare

**add** [y\_coordinate], **ax**

**call** drawsquare

**pop** **ax**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackSPiece\_2

**proc** blackZPiece\_1

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **ax**

        ; OO

        ;  OO

**mov** [light\_colour], 0h ; blacks

**mov** [main\_colour], 0h

**mov** [border\_colour], 0h

**mov** **ax**, [square\_size] ; mov square size to a register

**call** drawsquare

**add** [x\_coordinate], **ax**

**call** drawsquare

**add** [y\_coordinate], **ax**

**call** drawsquare

**add** [x\_coordinate], **ax**

**call** drawsquare

**pop** **ax**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackZPiece\_1

**proc** blackZPiece\_2

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **ax**

        ;  O

        ; OO

        ; O

**mov** [light\_colour], 0h ; reds

**mov** [main\_colour], 0h

**mov** [border\_colour], 0h

**mov** **ax**, [square\_size] ; mov square size to a register

**add** [x\_coordinate], **ax**

**call** drawsquare

**add** [y\_coordinate], **ax**

**call** drawsquare

**sub** [x\_coordinate], **ax**

**call** drawsquare

**add** [y\_coordinate], **ax**

**call** drawsquare

**pop** **ax**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** blackZPiece\_2

## לוגיקות

### proc intializeRandom

**תיאור הפרוצדורה**פרוצדורה זו מאתחלת את הזרע. צריך לקרוא לפרוצדורה זו רק פעם אחת בתחילת הקוד

**משתנים**seed – הזרע של הראנדום

**קוד הפרוצדורה**

**proc** initializeRandom

    ; This proc doesn't get any value

    ; This proc will set the modulus, multplier, increment and seed

    ; al = modulus

    ; ah = multiplier

    ; bl = increment

    ; bh = seed

**pusha**

**mov** **ah**, 0h

**int** 1ah

**mov** **ax**, **dx**

**mov** **ah**, 0h

    ; getting the seed

**mov** **dh**, 0h

**mov** **cx**, **ax**

**mov** **ax**, [modulus]

**mov** **cx**, **dx**

**mov** **dx**, 0h

**div** **cx**

**mov** **dx**, 0h

**mov** **cx**, 2

**mul** **cx**

**mov** [seed], **ax**

**popa**

**ret**

**endp** initializeRandom

### proc randomNum

**תיאור הפרוצדורה**פרוצדורה זו מייצרת מספר רנדומלי ומחדשת את הסיד

**משתנים**top\_limit - מספר מעל המספר המקסימלי שאליו יכול להגיע המספר הרנדומלי  
rand\_num – המספר הרנדומלי שהתקבל

**קוד הפרוצדורה**

**proc** randomNum

**pusha**

    ; This proc generates random number between 0 and the number in the register cx

    ; The cx number must be under 99!

    ; cx = top boundry of the random number

    ;

    ; result:

    ;   dl = the random number

**mov** **cx**, [top\_limit]

**inc** **cx**

**mov** **ax**, [seed]

**mov** **cx**, [multiplier]

**mul** **cx**

**add** **ax**, [increment]

**mov** **cx**, [modulus]

**mov** **dx**, 0

**div** **cx**

**mov** [seed], **dx**

**mov** **ax**, **dx**

**mov** **dx**, 0

**mov** **cx**, [top\_limit]

**div** **cx**

**mov** [rand\_num], **dl**

**popa**

**ret**

**endp** randomNum

### proc generate\_last\_7\_queue

**תיאור הפרוצדורה**פרוצדורה זו מגדירה את 7 המקומות האחרונים בתור בתור 7 הטטרומינואים בצורה רנדומלית

**משתנים**min\_last\_7\_queue – התחתית של התור שעדיין לא הוגדר  
top\_limit - מספר מעל המספר המקסימלי שאליו יכול להגיע המספר הרנדומלי, משתנה לפי כמה מקומות כבר הוגדרו  
queue – התור

**קוד הפרוצדורה**

**proc** generate\_last\_7\_queue

    ; an official tetris random generator mechanism

    ; needs to load the queue every time with all of the 7 tetreminos

    ; in a random order

    ; this procedure does it

**mov** [min\_queue\_last\_7], 14

**mov** **bx**, **offset** queue

**mov** **si**, 14

**mov** **cx**, 7

    reset\_last\_7\_loop: ; 100 is not an avaliable piece, so setting every spot we

                       ; want to change to 100 will let us know which spot we

                       ; already changed

**mov** [**bx+si**], 100

**add** **si**, 2

**loop** reset\_last\_7\_loop

**mov** **ax**, 0

**mov** **cx**, 7

    generate\_last\_7\_loop:

**mov** [top\_limit], **cx** ; generate a random location on the list

**call** randomnum

**mov** **dl**, [rand\_num]

**add** **dl**, [rand\_num]

**add** **dl**, [min\_queue\_last\_7]; the piece is a word, so si is doubled

**mov** **dh**, 0 ; now dx holds the position the position it wants to put a piece in

**mov** **si**, **dx**

        check\_if\_spot\_valid: ; check if the spot wasn't already taken

**cmp** [**bx+si**], 6

**ja** generate\_last\_7\_set ; when the spot is valid, continue

        ; if spot isnt valid:

**add** **si**, 2 ; try the one bove

**cmp** **si**, 28

**jb** check\_if\_spot\_valid ; if dx is still in range 14-27 check again

**mov** **dl**, [min\_queue\_last\_7] ; if it isn't start from the beginnig

**mov** **dh**, 0

**jmp** check\_if\_spot\_valid

        generate\_last\_7\_set:

**mov** [**bx+si**], **ax** ; when the spot is avaliable, put a piece in it

**inc** **ax** ; next piece

**cmp** [rand\_num], 0 ; if the chosen spot is the lowest avaliable spot, change the minimum to the next spot

**je** generate\_last\_7\_change\_min

**loop** generate\_last\_7\_loop

**ret**

        generate\_last\_7\_change\_min:

**add** [min\_queue\_last\_7], 2

**loop** generate\_last\_7\_loop

**ret**

**endp** generate\_last\_7\_queue

### פרוצדורות סיבוב

**תיאור הפרוצדורות**פרוצדורות אלו מסובבות את הטטרומינואים על גבי המסך

**משתנים**  
current\_piece – הטטרומינו הנוכחי, קובע איזה טטרומינו לסובב  
current\_piece\_rotation – הסיבוב של הטטרומינו הנוכחי, קובע איזו צורה של הטטרומינו לסובב  
x\_coordinate – קואורדינטת ה-x של הטטרומינו הנוכחי  
y\_coordinate – קואורדינטת ה-y של הטטרומינו הנוכחי  
pixelColour – הפונקציה צריכה לבדוק אם אפשר לסובב את הטטרומינו, משתנה זה מכיל את צבע הריבוע שנבדק

**קוד הפרוצדורות**

**proc** rotate\_left

**push** **ax**

**push** [x\_coordinate]

**push** [y\_coordinate]

**cmp** [current\_piece], 0 ; if t-piece

**je** rotate\_left\_t

**cmp** [current\_piece], 1 ; if o-piece

**je** rotate\_left\_o

**cmp** [current\_piece], 2 ; if j-piece

**je** rotate\_left\_j

**cmp** [current\_piece], 3 ; if l-piece

**je** rotate\_left\_l

**cmp** [current\_piece], 4 ; if i-piece

**je** rotate\_left\_i

**cmp** [current\_piece], 5 ; if s-piece

**je** rotate\_left\_s

**cmp** [current\_piece], 6 ; if z-piece

**je** rotate\_left\_z

**jmp** rotate\_left\_end

    rotate\_left\_o: ; o-piece

**call** drawopiece

**jmp** rotate\_left\_end ; o-piece has only 1 rotation

    rotate\_left\_z: ; z-piece

**inc** [current\_piece\_rotation] ; next position

**cmp** [current\_piece\_rotation], 4 ; make it return to 1 after 2

**jna** rotate\_left\_z\_draw ; make it return to 1 after 4

**mov** [current\_piece\_rotation], 1 ; make it return to 1 after 2

    rotate\_left\_z\_draw: ; draw s-piece

**mov** **ax**, [square\_size] ; move square size to a register (used for checking )

**cmp** [current\_piece\_rotation], 1 ; rotate to first position

**je** rotate\_left\_z\_draw\_1

**cmp** [current\_piece\_rotation], 2 ; rotate to second positon

**je** rotate\_left\_z\_draw\_2

**cmp** [current\_piece\_rotation], 3 ; rotate to third position

**je** rotate\_left\_z\_draw\_3

**cmp** [current\_piece\_rotation], 4 ; rotate to fourth position

**je** rotate\_left\_z\_draw\_4

**jmp** rotate\_left\_end

    rotate\_left\_z\_draw\_1:

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**add** [x\_coordinate], **ax**

**call** blackZPiece\_2 ; delete fourth position

**sub** [x\_coordinate], **ax**

**call** drawzPiece\_1 ; draw first position

**jmp** rotate\_left\_end

    rotate\_left\_z\_draw\_2:

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**call** blackZPiece\_1 ; delete fourth position

**call** drawzPiece\_2 ; draw first position

**jmp** rotate\_left\_end

    rotate\_left\_z\_draw\_3:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** blackZPiece\_2 ; delete first position

**add** [y\_coordinate], **ax**

**call** drawzPiece\_1 ; draw second position

**sub** [y\_coordinate], **ax**

**jmp** rotate\_left\_end

    rotate\_left\_z\_draw\_4:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** blackZPiece\_1 ; delete third position

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** drawzPiece\_2 ; draw fourth position

**sub** [x\_coordinate], **ax**

**jmp** rotate\_left\_end

    rotate\_left\_s: ; s-piece

**inc** [current\_piece\_rotation] ; next position

**cmp** [current\_piece\_rotation], 4 ; make it return to 1 after 2

**jna** rotate\_left\_s\_draw ; make it return to 1 after 4

**mov** [current\_piece\_rotation], 1 ; make it return to 1 after 2

    rotate\_left\_s\_draw: ; draw s-piece

**mov** **ax**, [square\_size] ; move square size to a register (used for checking )

**cmp** [current\_piece\_rotation], 1 ; rotate to first position

**je** rotate\_left\_s\_draw\_1

**cmp** [current\_piece\_rotation], 2 ; rotate to second positon

**je** rotate\_left\_s\_draw\_2

**cmp** [current\_piece\_rotation], 3 ; rotate to third position

**je** rotate\_left\_s\_draw\_3

**cmp** [current\_piece\_rotation], 4 ; rotate to fourth position

**je** rotate\_left\_s\_draw\_4

**jmp** rotate\_left\_end

    rotate\_left\_s\_draw\_1:

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** blackSPiece\_2 ; delete fourth position

**sub** [x\_coordinate], **ax**

**call** drawSPiece\_1 ; draw first position

**jmp** rotate\_left\_end

    rotate\_left\_s\_draw\_2:

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** blackSPiece\_1 ; delete first position

**call** drawSPiece\_2 ; draw second position

**jmp** rotate\_left\_end

    rotate\_left\_s\_draw\_3:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** blackSPiece\_2 ; delete second position

**add** [y\_coordinate], **ax**

**call** drawSPiece\_1 ; draw third position

**sub** [y\_coordinate], **ax**

**jmp** rotate\_left\_end

    rotate\_left\_s\_draw\_4:

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** blackSPiece\_1 ; delete third position

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** drawSPiece\_2 ; draw fourth position

**sub** [x\_coordinate], **ax**

**jmp** rotate\_left\_end

    rotate\_left\_i: ; i-piece

**inc** [current\_piece\_rotation] ; next position

**cmp** [current\_piece\_rotation], 4 ; make it return to 1 after 2

**jna** rotate\_left\_i\_draw ; make it return to 1 after 4

**mov** [current\_piece\_rotation], 1 ; make it return to 1 after 2

    rotate\_left\_i\_draw: ; draw i-piece

**mov** **ax**, [square\_size] ; move square size to a register (used for checking )

**cmp** [current\_piece\_rotation], 1 ; rotate to first position

**je** rotate\_left\_i\_draw\_1

**cmp** [current\_piece\_rotation], 2 ; rotate to second positon

**je** rotate\_left\_i\_draw\_2

**cmp** [current\_piece\_rotation], 3 ; rotate to third position

**je** rotate\_left\_i\_draw\_3

**cmp** [current\_piece\_rotation], 4 ; rotate to fourth position

**je** rotate\_left\_i\_draw\_4

**jmp** rotate\_left\_end

    rotate\_left\_i\_draw\_1:

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #3

**jne** rotate\_left\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** blackipiece\_2 ; delete fourth position

**sub** [x\_coordinate], **ax**

**call** drawipiece\_1 ; draw first position

**jmp** rotate\_left\_end

    rotate\_left\_i\_draw\_2:

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #3

**jne** rotate\_left\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** blackipiece\_1 ; delete first position

**call** drawipiece\_2 ; draw second position

**jmp** rotate\_left\_end

    rotate\_left\_i\_draw\_3:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #3

**jne** rotate\_left\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** blackipiece\_2 ; delete second position

**add** [y\_coordinate], **ax**

**call** drawipiece\_1 ; draw third position

**sub** [y\_coordinate], **ax**

**jmp** rotate\_left\_end

    rotate\_left\_i\_draw\_4:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #3

**jne** rotate\_left\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** blackipiece\_1 ; delete third position

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** drawipiece\_2 ; draw fourth position

**sub** [x\_coordinate], **ax**

**jmp** rotate\_left\_end

    rotate\_left\_l: ; l-piece

**inc** [current\_piece\_rotation] ; next position

**cmp** [current\_piece\_rotation], 5 ; make it return to 1 after 4

**jne** rotate\_left\_l\_draw ; make it return to 1 after 4

**mov** [current\_piece\_rotation], 1 ; make it return to 1 after 4

    rotate\_left\_l\_draw: ; draw l-piece

**mov** **ax**, [square\_size] ; move square size to a register (used for checking )

**cmp** [current\_piece\_rotation], 1 ; rotate to first position

**je** rotate\_left\_l\_draw\_1

**cmp** [current\_piece\_rotation], 2 ; rotate to second positon

**je** rotate\_left\_l\_draw\_2

**cmp** [current\_piece\_rotation], 3 ; rotate to third positon

**je** rotate\_left\_l\_draw\_3

**cmp** [current\_piece\_rotation], 4 ; rotate to 4rth positon

**je** rotate\_left\_l\_draw\_4

**jmp** rotate\_left\_end

    rotate\_left\_l\_draw\_1:

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**call** blacklpiece\_4 ; delete fourth position

**call** drawlpiece\_1 ; draw first position

**jmp** rotate\_left\_end

    rotate\_left\_l\_draw\_2:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** blacklpiece\_1 ; delete fourth position

**call** drawlpiece\_2 ; draw first position

**jmp** rotate\_left\_end

    rotate\_left\_l\_draw\_3:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**sub** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** blacklpiece\_2 ; delete fourth position

**call** drawlpiece\_3 ; draw first position

**jmp** rotate\_left\_end

    rotate\_left\_l\_draw\_4:

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**call** blacklpiece\_3 ; delete fourth position

**call** drawlpiece\_4 ; draw first position

**jmp** rotate\_left\_end

    rotate\_left\_j: ; j-piece

**inc** [current\_piece\_rotation] ; next position

**cmp** [current\_piece\_rotation], 5 ; make it return to 1 after 4

**jne** rotate\_left\_j\_draw ; make it return to 1 after 4

**mov** [current\_piece\_rotation], 1 ; make it return to 1 after 4

    rotate\_left\_j\_draw: ; draw t-piece

**mov** **ax**, [square\_size] ; move square size to a register (used for checking )

**cmp** [current\_piece\_rotation], 1 ; rotate to first position

**je** rotate\_left\_j\_draw\_1

**cmp** [current\_piece\_rotation], 2 ; rotate to second positon

**je** rotate\_left\_j\_draw\_2

**cmp** [current\_piece\_rotation], 3 ; rotate to third positon

**je** rotate\_left\_j\_draw\_3

**cmp** [current\_piece\_rotation], 4 ; rotate to 4rth positon

**je** rotate\_left\_j\_draw\_4

**jmp** rotate\_left\_end

    rotate\_left\_j\_draw\_1: ; rotate to first position

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**call** blackjpiece\_4 ; delete fourth position

**call** drawjpiece\_1 ; draw first position

**jmp** rotate\_left\_end

    rotate\_left\_j\_draw\_2: ; rotate to first position

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**call** blackjpiece\_1 ; delete fourth position

**call** drawjpiece\_2 ; draw first position

**jmp** rotate\_left\_end

    rotate\_left\_j\_draw\_3: ; rotate to first position

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** blackjpiece\_2 ; delete fourth position

**call** drawjpiece\_3 ; draw first position

**jmp** rotate\_left\_end

    rotate\_left\_j\_draw\_4: ; rotate to first position

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_left\_fail

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_left\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** blackjpiece\_3 ; delete fourth position

**call** drawjpiece\_4 ; draw first position

**jmp** rotate\_left\_end

    rotate\_left\_t: ; t-piece

**inc** [current\_piece\_rotation] ; next position

**cmp** [current\_piece\_rotation], 5 ; make it return to 1 after 4

**jne** rotate\_left\_t\_draw ; make it return to 1 after 4

**mov** [current\_piece\_rotation], 1 ; make it return to 1 after 4

    rotate\_left\_t\_draw: ; draw j-piece

**mov** **ax**, [square\_size] ; move square size to a register (used for checking)

**cmp** [current\_piece\_rotation], 1 ; rotate to first position

**je** rotate\_left\_t\_draw\_1

**cmp** [current\_piece\_rotation], 2 ; rotate to second positon

**je** rotate\_left\_t\_draw\_2

**cmp** [current\_piece\_rotation], 3 ; rotate to third positon

**je** rotate\_left\_t\_draw\_3

**cmp** [current\_piece\_rotation], 4 ; rotate to 4rth positon

**je** rotate\_left\_t\_draw\_4

**jmp** rotate\_left\_end

    rotate\_left\_t\_draw\_1: ; rotate to first position

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate

**jne** rotate\_left\_fail

**sub** [y\_coordinate], **ax**

**call** blacktpiece\_4 ; delete fourth position

**call** drawtpiece\_1 ; draw first position

**jmp** rotate\_left\_end

    rotate\_left\_t\_draw\_2: ; rotate to second position

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate

**jne** rotate\_left\_fail

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** blacktpiece\_1 ; delete first position

**call** drawtpiece\_2 ; draw second position

**jmp** rotate\_left\_end

    rotate\_left\_t\_draw\_3: ; rotate to third position

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate

**jne** rotate\_left\_fail

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** blacktpiece\_2 ; delete second position

**call** drawtpiece\_3 ; draw third position

**jmp** rotate\_left\_end

    rotate\_left\_t\_draw\_4: ; rotate to fourth position

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate

**jne** rotate\_left\_fail

**sub** [x\_coordinate], **ax**

**call** blacktpiece\_3 ; delete third position

**call** drawtpiece\_4 ; draw fourth position

**jmp** rotate\_left\_end

    rotate\_left\_fail:

**cmp** [current\_piece\_rotation], 1

**je** rotate\_left\_fail\_1

**dec** [current\_piece\_rotation]

**jmp** rotate\_left\_end

    rotate\_left\_fail\_1:

**mov** [current\_piece\_rotation], 4

    rotate\_left\_end:

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**pop** **ax**

**ret**

**endp** rotate\_left

**proc** rotate\_right

**push** **ax**

**push** [x\_coordinate]

**push** [y\_coordinate]

**cmp** [current\_piece], 0 ; if t-piece

**je** rotate\_right\_t

**cmp** [current\_piece], 1 ; if o-piece

**je** rotate\_right\_o

**cmp** [current\_piece], 2 ; if j-piece

**je** rotate\_right\_j

**cmp** [current\_piece], 3 ; if l-piece

**je** rotate\_right\_l

**cmp** [current\_piece], 4 ; if i-piece

**je** rotate\_right\_i

**cmp** [current\_piece], 5 ; if s-piece

**je** rotate\_right\_s

**cmp** [current\_piece], 6 ; if z-piece

**je** rotate\_right\_z

**jmp** rotate\_right\_end

    rotate\_right\_o: ; o-piece

**call** drawopiece

**jmp** rotate\_right\_end ; o-piece has only 1 rotation

    rotate\_right\_z: ; s-piece

**dec** [current\_piece\_rotation] ; next position

**cmp** [current\_piece\_rotation], 1 ; make it return to 1 after 2

**jnb** rotate\_right\_z\_draw ; make it return to 1 after 4

**mov** [current\_piece\_rotation], 4 ; make it return to 1 after 2

    rotate\_right\_z\_draw: ; draw s-piece

**mov** **ax**, [square\_size] ; move square size to a register (used for checking )

**cmp** [current\_piece\_rotation], 1 ; rotate to first position

**je** rotate\_right\_z\_draw\_1

**cmp** [current\_piece\_rotation], 2 ; rotate to second positon

**je** rotate\_right\_z\_draw\_2

**cmp** [current\_piece\_rotation], 3 ; rotate to third positon

**je** rotate\_right\_z\_draw\_3

**cmp** [current\_piece\_rotation], 4 ; rotate to fourth positon

**je** rotate\_right\_z\_draw\_4

**jmp** rotate\_right\_end

    rotate\_right\_z\_draw\_1:

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** blackzPiece\_2 ; delete second position

**call** drawzPiece\_1 ; draw first position

**jmp** rotate\_right\_end

    rotate\_right\_z\_draw\_2:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**add** [y\_coordinate], **ax**

**call** blackzPiece\_1 ; delete third position

**sub** [y\_coordinate], **ax**

**call** drawzPiece\_2 ; draw second position

**jmp** rotate\_right\_end

    rotate\_right\_z\_draw\_3:

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** blackzPiece\_2 ; delete fourth position

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** drawzPiece\_1 ; draw third position

**sub** [y\_coordinate], **ax**

**jmp** rotate\_right\_end

    rotate\_right\_z\_draw\_4:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** blackzPiece\_1 ; delete first position

**add** [x\_coordinate], **ax**

**call** drawzPiece\_2 ; draw fourth position

**sub** [x\_coordinate], **ax**

**jmp** rotate\_right\_end

    rotate\_right\_s: ; s-piece

**dec** [current\_piece\_rotation] ; next position

**cmp** [current\_piece\_rotation], 1 ; make it return to 1 after 2

**jnb** rotate\_right\_s\_draw ; make it return to 1 after 4

**mov** [current\_piece\_rotation], 4 ; make it return to 1 after 2

    rotate\_right\_s\_draw: ; draw s-piece

**mov** **ax**, [square\_size] ; move square size to a register (used for checking )

**cmp** [current\_piece\_rotation], 1 ; rotate to first position

**je** rotate\_right\_s\_draw\_1

**cmp** [current\_piece\_rotation], 2 ; rotate to second positon

**je** rotate\_right\_s\_draw\_2

**cmp** [current\_piece\_rotation], 3 ; rotate to third positon

**je** rotate\_right\_s\_draw\_3

**cmp** [current\_piece\_rotation], 4 ; rotate to fourth positon

**je** rotate\_right\_s\_draw\_4

**jmp** rotate\_right\_end

    rotate\_right\_s\_draw\_1:

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**call** blackSPiece\_2 ; delete second position

**call** drawSPiece\_1 ; draw first position

**jmp** rotate\_right\_end

    rotate\_right\_s\_draw\_2:

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**add** [y\_coordinate], **ax**

**call** blackSPiece\_1 ; delete third position

**sub** [y\_coordinate], **ax**

**call** drawSPiece\_2 ; draw second position

**jmp** rotate\_right\_end

    rotate\_right\_s\_draw\_3:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** blackSPiece\_2 ; delete fourth position

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** drawSPiece\_1 ; draw third position

**sub** [y\_coordinate], **ax**

**jmp** rotate\_right\_end

    rotate\_right\_s\_draw\_4:

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** blackSPiece\_1 ; delete fourth position

**add** [x\_coordinate], **ax**

**call** drawSPiece\_2 ; draw third position

**sub** [x\_coordinate], **ax**

**jmp** rotate\_right\_end

    rotate\_right\_i: ; i-piece

**dec** [current\_piece\_rotation] ; next position

**cmp** [current\_piece\_rotation], 1 ; make it return to 1 after 2

**jnb** rotate\_right\_i\_draw ; make it return to 1 after 4

**mov** [current\_piece\_rotation], 4 ; make it return to 1 after 2

    rotate\_right\_i\_draw: ; draw i-piece

**mov** **ax**, [square\_size] ; move square size to a register (used for checking )

**cmp** [current\_piece\_rotation], 1 ; rotate to first position

**je** rotate\_right\_i\_draw\_1

**cmp** [current\_piece\_rotation], 2 ; rotate to second positon

**je** rotate\_right\_i\_draw\_2

**cmp** [current\_piece\_rotation], 3 ; rotate to third positon

**je** rotate\_right\_i\_draw\_3

**cmp** [current\_piece\_rotation], 4 ; rotate to fourth positon

**je** rotate\_right\_i\_draw\_4

**jmp** rotate\_right\_end

    rotate\_right\_i\_draw\_1:

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #3

**jne** rotate\_right\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** blackipiece\_2 ; delete second position

**call** drawipiece\_1 ; draw first position

**jmp** rotate\_right\_end

    rotate\_right\_i\_draw\_2:

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #3

**jne** rotate\_right\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** blackipiece\_1 ; delete third position

**sub** [y\_coordinate], **ax**

**call** drawipiece\_2 ; draw second position

**jmp** rotate\_right\_end

    rotate\_right\_i\_draw\_3:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #3

**jne** rotate\_right\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** blackipiece\_2 ; delete fourth position

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** drawipiece\_1 ; draw third position

**sub** [y\_coordinate], **ax**

**jmp** rotate\_right\_end

    rotate\_right\_i\_draw\_4:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #3

**jne** rotate\_right\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** blackipiece\_1 ; delete first position

**add** [x\_coordinate], **ax**

**call** drawipiece\_2 ; draw fourth position

**sub** [x\_coordinate], **ax**

**jmp** rotate\_right\_end

    rotate\_right\_l: ; l-piece

**dec** [current\_piece\_rotation] ; previouse position

**cmp** [current\_piece\_rotation], 0 ; make it return to 4 after 1

**jne** rotate\_right\_l\_draw ; make it return to 4 after 1

**mov** [current\_piece\_rotation], 4 ; make it return to 4 after 1

    rotate\_right\_l\_draw: ; draw l-piece

**mov** **ax**, [square\_size] ; move square size to a register (used for checking )

**cmp** [current\_piece\_rotation], 1 ; rotate to first position

**je** rotate\_right\_l\_draw\_1

**cmp** [current\_piece\_rotation], 2 ; rotate to second positon

**je** rotate\_right\_l\_draw\_2

**cmp** [current\_piece\_rotation], 3 ; rotate to third positon

**je** rotate\_right\_l\_draw\_3

**cmp** [current\_piece\_rotation], 4 ; rotate to 4rth positon

**je** rotate\_right\_l\_draw\_4

**jmp** rotate\_right\_fail

    rotate\_right\_l\_draw\_1:

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**call** blacklpiece\_2 ; delete fourth position

**call** drawlpiece\_1 ; draw first position

**jmp** rotate\_right\_end

    rotate\_right\_l\_draw\_2:

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**call** blacklpiece\_3 ; delete fourth position

**call** drawlpiece\_2 ; draw first position

**jmp** rotate\_right\_end

    rotate\_right\_l\_draw\_3:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** blacklpiece\_4 ; delete fourth position

**call** drawlpiece\_3 ; draw first position

**jmp** rotate\_right\_end

    rotate\_right\_l\_draw\_4:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**sub** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** blacklpiece\_1 ; delete fourth position

**call** drawlpiece\_4 ; draw first position

**jmp** rotate\_right\_end

    rotate\_right\_j: ; j-piece

**dec** [current\_piece\_rotation] ; previouse position

**cmp** [current\_piece\_rotation], 0 ; make it return to 4 after 1

**jne** rotate\_right\_j\_draw ; make it return to 4 after 1

**mov** [current\_piece\_rotation], 4 ; make it return to 4 after 1

    rotate\_right\_j\_draw: ; draw j-piece

**mov** **ax**, [square\_size] ; move square size to a register (used for checking )

**cmp** [current\_piece\_rotation], 1 ; rotate to first position

**je** rotate\_right\_j\_draw\_1

**cmp** [current\_piece\_rotation], 2 ; rotate to second positon

**je** rotate\_right\_j\_draw\_2

**cmp** [current\_piece\_rotation], 3 ; rotate to third positon

**je** rotate\_right\_j\_draw\_3

**cmp** [current\_piece\_rotation], 4 ; rotate to 4rth positon

**je** rotate\_right\_j\_draw\_4

**jmp** rotate\_right\_fail

    rotate\_right\_j\_draw\_1:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** blackjpiece\_2 ; delete fourth position

**call** drawjpiece\_1 ; draw first position

**jmp** rotate\_right\_end

    rotate\_right\_j\_draw\_2:

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [x\_coordinate], **ax** ; return to cursor's position

**call** blackjpiece\_3 ; delete fourth position

**call** drawjpiece\_2 ; draw first position

**jmp** rotate\_right\_end

    rotate\_right\_j\_draw\_3:

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**call** blackjpiece\_4 ; delete fourth position

**call** drawjpiece\_3 ; draw first position

**jmp** rotate\_right\_end

    rotate\_right\_j\_draw\_4:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #1

**jne** rotate\_right\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate #2

**jne** rotate\_right\_fail

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** blackjpiece\_1 ; delete fourth position

**call** drawjpiece\_4 ; draw first position

**jmp** rotate\_right\_end

    rotate\_right\_t: ; t-piece

**dec** [current\_piece\_rotation] ; previouse position

**cmp** [current\_piece\_rotation], 0 ; make it return to 4 after 1

**jne** rotate\_right\_t\_draw ; make it return to 4 after 1

**mov** [current\_piece\_rotation], 4 ; make it return to 4 after 1

    rotate\_right\_t\_draw: ; draw t-piece

**mov** **ax**, [square\_size] ; move square size to a register (used for checking )

**cmp** [current\_piece\_rotation], 1 ; rotate to first position

**je** rotate\_right\_t\_draw\_1

**cmp** [current\_piece\_rotation], 2 ; rotate to second positon

**je** rotate\_right\_t\_draw\_2

**cmp** [current\_piece\_rotation], 3 ; rotate to third positon

**je** rotate\_right\_t\_draw\_3

**cmp** [current\_piece\_rotation], 4 ; rotate to 4rth positon

**je** rotate\_right\_t\_draw\_4

**jmp** rotate\_right\_fail

    rotate\_right\_t\_draw\_1: ; rotate to first position ;y+1

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate

**jne** rotate\_right\_fail

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** blacktpiece\_2 ; delete second position

**call** drawtpiece\_1 ; draw first position

**jmp** rotate\_right\_end

    rotate\_right\_t\_draw\_2: ; rotate to second position

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate

**jne** rotate\_right\_fail

**sub** [x\_coordinate], **ax**

**call** blacktpiece\_3 ; delete third position

**call** drawtpiece\_2 ; draw second position

**jmp** rotate\_right\_end

    rotate\_right\_t\_draw\_3: ; rotate to third position

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate

**jne** rotate\_right\_fail

**sub** [y\_coordinate], **ax**

**call** blacktpiece\_4 ; delete fourth position

**call** drawtpiece\_3 ; draw third position

**jmp** rotate\_right\_end

    rotate\_right\_t\_draw\_4: ; rotate to fourth position

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; if the square that wasn't occupied is already occupied, don't rotate

**jne** rotate\_right\_fail

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** blacktpiece\_1 ; delete first position

**call** drawtpiece\_4 ; draw fourth position

**jmp** rotate\_right\_end

    rotate\_right\_fail:

**cmp** [current\_piece\_rotation], 4

**je** rotate\_right\_fail\_1

**inc** [current\_piece\_rotation]

**jmp** rotate\_right\_end

    rotate\_right\_fail\_1:

**mov** [current\_piece\_rotation], 1

    rotate\_right\_end:

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**pop** **ax**

**ret**

**endp** rotate\_right

### פרוצדורות תזוזה

**תיאור הפרוצדורות**פרוצדורות אלו מזיזות את הטטרומינואים על גבי המסך

**משתנים**  
current\_piece – הטטרומינו הנוכחי, קובע איזה טטרומינו להזיז  
current\_piece\_rotation – הסיבוב של הטטרומינו הנוכחי, קובע איזו צורה של הטטרומינו להזיז  
x\_coordinate – קואורדינטת ה-x של הטטרומינו הנוכחי  
y\_coordinate – קואורדינטת ה-y של הטטרומינו הנוכחי  
move\_down\_failed – האם הריבוע הצליח לרדת למטה  
pixelColour – הפונקציה צריכה לבדוק אם אפשר להזיז את הטטרומינו, משתנה זה מכיל את צבע הריבוע שנבדק

**קוד הפרוצדורות**

**proc** move\_left

**push** **ax**

**push** [y\_coordinate]

**push** [x\_coordinate]

**mov** **ax**, [square\_size] ; square size in a register

**cmp** [current\_piece], 0

**je** move\_left\_t ; if t-piece

**cmp** [current\_piece], 1

**je** move\_left\_o ; if o-piece

**cmp** [current\_piece], 2

**je** move\_left\_j ; if j-piece

**cmp** [current\_piece], 3

**je** move\_left\_l ; if l-piece

**cmp** [current\_piece], 4

**je** move\_left\_i ; if i-piece

**cmp** [current\_piece], 5

**je** move\_left\_s ; if s-piece

**cmp** [current\_piece], 6

**je** move\_left\_z ; if z-piece

**jmp** move\_left\_end

    move\_left\_o:

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackopiece ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawopiece ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_z:

**cmp** [current\_piece\_rotation], 1 ; different actions based on rotation

**je** move\_left\_z\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_left\_z\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_left\_z\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_left\_z\_4

**jmp** move\_left\_end

    move\_left\_z\_1:

**sub** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackZPiece\_1 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawZPiece\_1 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_z\_2:

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 3

**jne** move\_left\_end

**add** [x\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackZPiece\_2 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawZPiece\_2 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_z\_3:

**add** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 3

**jne** move\_left\_end

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**add** [y\_coordinate], **ax**

**call** blackZPiece\_1 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawZPiece\_1 ; redraw it one square left

**sub** [y\_coordinate], **ax**

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_z\_4:

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 3

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**add** [x\_coordinate], **ax**

**call** blackZPiece\_2 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawZPiece\_2 ; redraw it one square left

**sub** [x\_coordinate], **ax**

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_s:

**cmp** [current\_piece\_rotation], 1 ; different actions based on rotation

**je** move\_left\_s\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_left\_s\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_left\_s\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_left\_s\_4

**jmp** move\_left\_end

    move\_left\_s\_1:

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [x\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackSPiece\_1 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawSPiece\_1 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_s\_2:

**sub** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 3

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackSPiece\_2 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawSPiece\_2 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_s\_3:

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**add** [y\_coordinate], **ax**

**call** blackSPiece\_1 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawSPiece\_1 ; redraw it one square left

**sub** [y\_coordinate], **ax**

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_s\_4:

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**add** [x\_coordinate], **ax**

**call** blackSPiece\_2 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawSPiece\_2 ; redraw it one square left

**sub** [x\_coordinate], **ax**

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_i:

**cmp** [current\_piece\_rotation], 1 ; different actions based on rotation

**je** move\_left\_i\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_left\_i\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_left\_i\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_left\_i\_4

**jmp** move\_left\_end

    move\_left\_i\_1:

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [x\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackipiece\_1 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawipiece\_1 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_i\_2:

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 3

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 4

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackipiece\_2 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawipiece\_2 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_i\_3:

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [x\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**add** [y\_coordinate], **ax**

**call** blackipiece\_1 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawipiece\_1 ; redraw it one square left

**sub** [y\_coordinate], **ax**

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_i\_4:

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 3

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 4

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**add** [x\_coordinate], **ax**

**call** blackipiece\_2 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawipiece\_2 ; redraw it one square left

**sub** [x\_coordinate], **ax**

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_l:

**cmp** [current\_piece\_rotation], 1 ; different actions based on rotation

**je** move\_left\_l\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_left\_l\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_left\_l\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_left\_l\_4

**jmp** move\_left\_end

    move\_left\_l\_1:

**sub** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

        ; l-piece 1st position has no third row so there's no point in checking it

**sub** [y\_coordinate], **ax** ; return to cursor's position

**add** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacklpiece\_1 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawlpiece\_1 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_l\_2:

**sub** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 3

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacklpiece\_2 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawlpiece\_2 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_l\_3:

        ; l-piece 1st position has no first row so there's no point in checking it

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 3

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacklpiece\_3 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawlpiece\_3 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_l\_4:

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 3

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacklpiece\_4 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawlpiece\_4 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_j:

**cmp** [current\_piece\_rotation], 1 ; different actions based on rotation

**je** move\_left\_j\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_left\_j\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_left\_j\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_left\_j\_4

**jmp** move\_left\_end

    move\_left\_j\_1:

**sub** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

        ; j-piece 1st position has no third row so there's no point in checking it

**sub** [x\_coordinate], **ax** ; return to cursor's position

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackjpiece\_1 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawjpiece\_1 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_j\_2:

**sub** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 3

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackjpiece\_2 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawjpiece\_2 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_j\_3:

        ; j-piece 1st position has no first row so there's no point in checking it

**add** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 3

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax** ; return to cursor position

**add** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackjpiece\_3 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawjpiece\_3 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_j\_4:

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackjpiece\_4 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawjpiece\_4 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_t:

**cmp** [current\_piece\_rotation], 1 ; different actions based on rotation

**je** move\_left\_t\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_left\_t\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_left\_t\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_left\_t\_4

**jmp** move\_left\_end

    move\_left\_t\_1:

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

        ; t-piece 1st position has no first row so there's no point in checking it

**sub** [y\_coordinate], **ax** ; return to cursor position

**add** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacktpiece\_1 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawtpiece\_1 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_t\_2:

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 3

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacktpiece\_2 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawtpiece\_2 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_t\_3:

        ; t-piece 3rd position has no first row so there's no point in checking it

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 3

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacktpiece\_3 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawtpiece\_3 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_t\_4:

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 1

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 2

**jne** move\_left\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move left if it's blocked - row 3

**jne** move\_left\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacktpiece\_4 ; erase piece

**sub** [x\_coordinate], **ax**

**call** drawtpiece\_4 ; redraw it one square left

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_left\_end

    move\_left\_end:

**pop** [x\_coordinate]

**pop** [y\_coordinate]

**pop** **ax**

**ret**

**endp** move\_left

**proc** move\_right

**push** **ax**

**push** [y\_coordinate]

**push** [x\_coordinate]

**mov** **ax**, [square\_size] ; square size in a register

**cmp** [current\_piece], 0

**je** move\_right\_t ; t-piece

**cmp** [current\_piece], 1

**je** move\_right\_o ; o-piece

**cmp** [current\_piece], 2

**je** move\_right\_j ; j-piece

**cmp** [current\_piece], 3

**je** move\_right\_l ; l-piece

**cmp** [current\_piece], 4

**je** move\_right\_i ; i-piece

**cmp** [current\_piece], 5

**je** move\_right\_s ; s-piece

**cmp** [current\_piece], 6

**je** move\_right\_z ; z-piece

**jmp** move\_right\_end

    move\_right\_o:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackopiece ; erase piece

**add** [x\_coordinate], **ax**

**call** drawopiece ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

        move\_right\_z:

**cmp** [current\_piece\_rotation], 1

**je** move\_right\_z\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_right\_z\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_right\_z\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_right\_z\_4

**jmp** move\_right\_end

    move\_right\_z\_1:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackzPiece\_1 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawzPiece\_1 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_z\_2:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackzPiece\_2 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawzPiece\_2 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_z\_3:

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackzPiece\_1 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawzPiece\_1 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**sub** [y\_coordinate], **ax**

**jmp** move\_right\_end

    move\_right\_z\_4:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**add** [x\_coordinate], **ax**

**call** blackzPiece\_2 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawzPiece\_2 ; redraw it one square right

**sub** [x\_coordinate], **ax**

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_s:

**cmp** [current\_piece\_rotation], 1

**je** move\_right\_s\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_right\_s\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_right\_s\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_right\_s\_4

**jmp** move\_right\_end

    move\_right\_s\_1:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackSPiece\_1 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawSPiece\_1 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_s\_2:

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 3

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackSPiece\_2 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawSPiece\_2 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_s\_3:

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 3

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**add** [y\_coordinate], **ax**

**call** blackSPiece\_1 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawSPiece\_1 ; redraw it one square right

**sub** [y\_coordinate], **ax**

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_s\_4:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 3

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**add** [x\_coordinate], **ax**

**call** blackSPiece\_2 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawSPiece\_2 ; redraw it one square right

**sub** [x\_coordinate], **ax**

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_i:

**cmp** [current\_piece\_rotation], 1

**je** move\_right\_i\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_right\_i\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_right\_i\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_right\_i\_4

**jmp** move\_right\_end

    move\_right\_i\_1:

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackipiece\_1 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawipiece\_1 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_i\_2:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 3

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 4

**jne** move\_right\_end

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackipiece\_2 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawipiece\_2 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_i\_3:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**add** [y\_coordinate], **ax**

**call** blackipiece\_1 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawipiece\_1 ; redraw it one square right

**sub** [y\_coordinate], **ax**

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_i\_4:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 3

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 4

**jne** move\_right\_end

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**add** [x\_coordinate], **ax**

**call** blackipiece\_2 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawipiece\_2 ; redraw it one square right

**sub** [x\_coordinate], **ax**

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_l:

**cmp** [current\_piece\_rotation], 1

**je** move\_right\_l\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_right\_l\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_right\_l\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_right\_l\_4

**jmp** move\_right\_end

    move\_right\_l\_1:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

        ; l-piece 1st position has no first row so there's no point in checking it

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacklpiece\_1 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawlpiece\_1 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_l\_2:

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 3

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacklpiece\_2 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawlpiece\_2 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_l\_3:

        ; l-piece 3rd position has no first row so there's no point in checking it

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 3

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacklpiece\_3 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawlpiece\_3 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_l\_4:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 3

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacklpiece\_4 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawlpiece\_4 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_j:

**cmp** [current\_piece\_rotation], 1

**je** move\_right\_j\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_right\_j\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_right\_j\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_right\_j\_4

**jmp** move\_right\_end

    move\_right\_j\_1:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

        ; j-piece 1st position has no first row so there's no point in checking it

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackjpiece\_1 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawjpiece\_1 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_j\_2:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 3

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackjpiece\_2 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawjpiece\_2 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_j\_3:

        ; j-piece 3rd position doesn't have a first row so there's no point in checking it

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 3

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackjpiece\_3 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawjpiece\_3 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_j\_4:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 3

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blackjpiece\_4 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawjpiece\_4 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_t:

**cmp** [current\_piece\_rotation], 1

**je** move\_right\_t\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_right\_t\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_right\_t\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_right\_t\_4

**jmp** move\_right\_end

    move\_right\_t\_1:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

        ; t-piece 1st position has no first row so there's no point in checking it

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacktpiece\_1 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawtpiece\_1 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_t\_2:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 3

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacktpiece\_2 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawtpiece\_2 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_t\_3:

        ; t-piece 3rd position has no first row so there's no point in checking it

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**sub** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 3

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacktpiece\_3 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawtpiece\_3 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_t\_4:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 1

**jne** move\_right\_end

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 2

**jne** move\_right\_end

**add** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move right if it's blocked - row 3

**jne** move\_right\_end

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [x\_coordinate] ; momenteraly pop x\_coordinate in order to permenantly change it

**call** blacktpiece\_4 ; erase piece

**add** [x\_coordinate], **ax**

**call** drawtpiece\_4 ; redraw it one square right

**push** [x\_coordinate] ; push x\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_right\_end

    move\_right\_end:

**pop** [x\_coordinate]

**pop** [y\_coordinate]

**pop** **ax**

**ret**

**endp** move\_right

**proc** move\_down

**push** **ax**

**push** [x\_coordinate]

**push** [y\_coordinate]

**mov** **ax**, [square\_size] ; square size in a register

**cmp** [current\_piece], 0

**je** move\_down\_t ; if t-piece

**cmp** [current\_piece], 1

**je** move\_down\_o ; if o-piece

**cmp** [current\_piece], 2

**je** move\_down\_j ; if j-piece

**cmp** [current\_piece], 3

**je** move\_down\_l ; if l-piece

**cmp** [current\_piece], 4

**je** move\_down\_i ; if i-piece

**cmp** [current\_piece], 5

**je** move\_down\_s ; if s-piece

**cmp** [current\_piece], 6

**je** move\_down\_z ; if z-piece

**jmp** move\_down\_end

    move\_down\_o:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blackopiece ; erase piece

**add** [y\_coordinate], **ax**

**call** drawopiece ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_z:

**cmp** [current\_piece\_rotation], 1 ; every rotation falls down differently

**je** move\_down\_z\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_down\_z\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_down\_z\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_down\_z\_4

**jmp** move\_down\_end

    move\_down\_z\_1:

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 3

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blackzPiece\_1 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawzPiece\_1 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_z\_2:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blackzPiece\_2 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawzPiece\_2 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_z\_3:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 3

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**add** [y\_coordinate], **ax**

**call** blackzPiece\_1 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawzPiece\_1 ; redraw it one square down

**sub** [y\_coordinate], **ax**

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_z\_4:

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blackzPiece\_2 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawzPiece\_2 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**sub** [x\_coordinate], **ax**

**jmp** move\_down\_end

    move\_down\_s:

**cmp** [current\_piece\_rotation], 1 ; every rotation falls down differently

**je** move\_down\_s\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_down\_s\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_down\_s\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_down\_s\_4

**jmp** move\_down\_end

    move\_down\_s\_1:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 3

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blackSPiece\_1 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawSPiece\_1 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_s\_2:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blackSPiece\_2 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawSPiece\_2 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_s\_3:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 3

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**add** [y\_coordinate], **ax**

**call** blackSPiece\_1 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawSPiece\_1 ; redraw it one square down

**sub** [y\_coordinate], **ax**

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_s\_4:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**add** [x\_coordinate], **ax**

**call** blackSPiece\_2 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawSPiece\_2 ; redraw it one square down

**sub** [x\_coordinate], **ax**

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_i:

**cmp** [current\_piece\_rotation], 1 ; every rotation falls down differently

**je** move\_down\_i\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_down\_i\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_down\_i\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_down\_i\_4

**jmp** move\_down\_end

    move\_down\_i\_1:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 3

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 4

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blackipiece\_1 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawipiece\_1 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_i\_2:

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blackipiece\_2 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawipiece\_2 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_i\_3:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 3

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 4

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**add** [y\_coordinate], **ax**

**call** blackipiece\_1 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawipiece\_1 ; redraw it one square down

**sub** [y\_coordinate], **ax**

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_i\_4:

**add** [x\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**add** [x\_coordinate], **ax**

**call** blackipiece\_2 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawipiece\_2 ; redraw it one square down

**sub** [x\_coordinate], **ax**

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_l:

**cmp** [current\_piece\_rotation], 1 ; every rotation falls down differently

**je** move\_down\_l\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_down\_l\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_down\_l\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_down\_l\_4

**jmp** move\_down\_end

    move\_down\_l\_1:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 3

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blacklpiece\_1 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawlpiece\_1 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_l\_2:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

        ; j-piece 2nd position doesn't have a 3rd column so there's no point in checking it

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blacklpiece\_2 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawlpiece\_2 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_l\_3:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blacklpiece\_3 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawlpiece\_3 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_l\_4:

        ; l-piece 4th position doesn't have a 1st column so there's no point in checking it

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blacklpiece\_4 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawlpiece\_4 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_j:

**cmp** [current\_piece\_rotation], 1 ; every rotation falls down differently

**je** move\_down\_j\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_down\_j\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_down\_j\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_down\_j\_4

**jmp** move\_down\_end

    move\_down\_j\_1:

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 3

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blackjpiece\_1 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawjpiece\_1 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_j\_2:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

        ; j-piece 2nd position doesn't have a 3rd column so there's no point in checking it

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blackjpiece\_2 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawjpiece\_2 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_j\_3:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 3

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blackjpiece\_3 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawjpiece\_3 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_j\_4:

        ; j-piece 4nd position doesn't have a 1st column so there's no point in checking it

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 3

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blackjpiece\_4 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawjpiece\_4 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_t:

**cmp** [current\_piece\_rotation], 1 ; every rotation falls down differently

**je** move\_down\_t\_1

**cmp** [current\_piece\_rotation], 2

**je** move\_down\_t\_2

**cmp** [current\_piece\_rotation], 3

**je** move\_down\_t\_3

**cmp** [current\_piece\_rotation], 4

**je** move\_down\_t\_4

**jmp** move\_down\_end

    move\_down\_t\_1:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 3

**jne** move\_down\_fail

**sub** [x\_coordinate], **ax** ; return to cursor position

**sub** [x\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blacktpiece\_1 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawtpiece\_1 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_t\_2:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

        ; t-piece 2nd position doesn't have a 3rc column so there's nothing to check there

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blacktpiece\_2 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawtpiece\_2 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_t\_3:

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 1

**jne** move\_down\_fail

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 3

**jne** move\_down\_fail

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blacktpiece\_3 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawtpiece\_3 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_t\_4:

        ; t-piece 2nd position doesn't have a 3rc column so there's nothing to check there

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 2

**jne** move\_down\_fail

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0 ; don't move down if it's blocked - column 3

**jne** move\_down\_fail

**sub** [y\_coordinate], **ax** ; return to cursor position

**sub** [y\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**sub** [x\_coordinate], **ax**

**pop** [y\_coordinate] ; momenteraly pop y\_coordinate in order to permenantly change it

**call** blacktpiece\_4 ; erase piece

**add** [y\_coordinate], **ax**

**call** drawtpiece\_4 ; redraw it one square down

**push** [y\_coordinate] ; push y\_coordinate back before jumping to the end to not mess up the pop at the end

**jmp** move\_down\_end

    move\_down\_fail:

**mov** [move\_down\_failed], 1

    move\_down\_end:

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**pop** **ax**

**ret**

**endp** move\_down

### proc generate\_piece

**תיאור הפרוצדורה**פרוצדורה זו מייצרת טטרומינו חדש  
x\_coordinate – קואורדינטת ה-x של הטטרומינו ליצירה  
y\_coordinate – קואורדינטת ה-y של הטטרומינו ליצירה

**משתנים**current\_piece – הטטרומינו לייצור

**קוד הפרוצדורה**

**proc** generate\_piece

**cmp** [current\_piece], 0 ; 0 = t-piece, 1 = o-piece, 2 = j-piece, 3 = l-piece, 4 = i-piece, 5 = s-piece, 6 = z-piece

**je** generate\_t

**cmp** [current\_piece], 1

**je** generate\_o

**cmp** [current\_piece], 2

**je** generate\_j

**cmp** [current\_piece], 3

**je** generate\_l

**cmp** [current\_piece], 4

**je** generate\_i

**cmp** [current\_piece], 5

**je** generate\_s

**cmp** [current\_piece], 6

**je** generate\_z

**ret**

    generate\_t:

**call** drawtpiece\_1

**ret**

    generate\_o:

**call** drawopiece

**ret**

    generate\_j:

**call** drawjpiece\_1

**ret**

    generate\_l:

**call** drawlpiece\_1

**ret**

    generate\_i:

**call** drawipiece\_1

**ret**

    generate\_s:

**call** drawspiece\_1

**ret**

    generate\_z:

**call** drawzpiece\_1

**ret**

**ret**

**endp** generate\_piece

### proc destoy\_piece

**תיאור הפרוצדורה**פרוצדורה זו מוחקת כל סוג של טטרומינו בכל צורה

**משתנים**current\_piece – הטטרומינו למחיקה  
current\_piece\_rotation – הסיבוב של הטטרומינו למחיקה  
x\_coordinate – קואורדינטת ה-x של הטטרומינו למחיקה  
y\_coordinate – קואורדינטת ה-y של הטטרומינו למחיקה

**קוד**

**proc** destroy\_piece

**cmp** [current\_piece], 0 ; 0 = t-piece, 1 = o-piece, 2 = j-piece, 3 = l-piece, 4 = i-piece, 5 = s-piece, 6 = z-piece

**je** destroy\_t

**cmp** [current\_piece], 1

**je** destroy\_o

**cmp** [current\_piece], 2

**je** destroy\_j

**cmp** [current\_piece], 3

**je** destroy\_l

**cmp** [current\_piece], 4

**je** destroy\_i

**cmp** [current\_piece], 5

**je** destroy\_s

**cmp** [current\_piece], 6

**je** destroy\_z

**ret**

    destroy\_t:

**cmp** [current\_piece\_rotation], 1

**je** destroy\_t\_1

**cmp** [current\_piece\_rotation], 2

**je** destroy\_t\_2

**cmp** [current\_piece\_rotation], 3

**je** destroy\_t\_3

**cmp** [current\_piece\_rotation], 4

**je** destroy\_t\_4

**ret**

        destroy\_t\_1:

**call** blacktpiece\_1

**ret**

        destroy\_t\_2:

**call** blacktpiece\_2

**ret**

        destroy\_t\_3:

**call** blacktpiece\_3

**ret**

        destroy\_t\_4:

**call** blacktpiece\_4

**ret**

    destroy\_o:

**call** blackopiece

**ret**

    destroy\_j:

**cmp** [current\_piece\_rotation], 1

**je** destroy\_j\_1

**cmp** [current\_piece\_rotation], 2

**je** destroy\_j\_2

**cmp** [current\_piece\_rotation], 3

**je** destroy\_j\_3

**cmp** [current\_piece\_rotation], 4

**je** destroy\_j\_4

**ret**

        destroy\_j\_1:

**call** blackjpiece\_1

**ret**

        destroy\_j\_2:

**call** blackjpiece\_2

**ret**

        destroy\_j\_3:

**call** blackjpiece\_3

**ret**

        destroy\_j\_4:

**call** blackjpiece\_4

**ret**

    destroy\_l:

**cmp** [current\_piece\_rotation], 1

**je** destroy\_l\_1

**cmp** [current\_piece\_rotation], 2

**je** destroy\_l\_2

**cmp** [current\_piece\_rotation], 3

**je** destroy\_l\_3

**cmp** [current\_piece\_rotation], 4

**je** destroy\_l\_4

**ret**

        destroy\_l\_1:

**call** blacklpiece\_1

**ret**

        destroy\_l\_2:

**call** blacklpiece\_2

**ret**

        destroy\_l\_3:

**call** blacklpiece\_3

**ret**

        destroy\_l\_4:

**call** blacklpiece\_4

**ret**

    destroy\_i:

**push** **ax**

**mov** **ax**, [square\_size]

**cmp** [current\_piece\_rotation], 1

**je** destroy\_i\_1

**cmp** [current\_piece\_rotation], 2

**je** destroy\_i\_2

**cmp** [current\_piece\_rotation], 3

**je** destroy\_i\_3

**cmp** [current\_piece\_rotation], 4

**je** destroy\_i\_4

**ret**

        destroy\_i\_1:

**call** blackipiece\_1

**pop** **ax**

**ret**

        destroy\_i\_2:

**call** blackipiece\_2

**pop** **ax**

**ret**

        destroy\_i\_3:

**push** **ax**

**mov** **ax**, [square\_size]

**add** [y\_coordinate], **ax**

**call** blackipiece\_1

**sub** [y\_coordinate], **ax**

**pop** **ax**

**ret**

        destroy\_i\_4:

**add** [x\_coordinate], **ax**

**call** blackipiece\_2

**sub** [x\_coordinate], **ax**

**pop** **ax**

**ret**

    destroy\_s:

**push** **ax**

**mov** **ax**, [square\_size]

**cmp** [current\_piece\_rotation], 1

**je** destroy\_s\_1

**cmp** [current\_piece\_rotation], 2

**je** destroy\_s\_2

**cmp** [current\_piece\_rotation], 3

**je** destroy\_s\_3

**cmp** [current\_piece\_rotation], 4

**je** destroy\_s\_4

**ret**

        destroy\_s\_1:

**call** blackspiece\_1

**pop** **ax**

**ret**

        destroy\_s\_2:

**call** blackspiece\_2

**pop** **ax**

**ret**

        destroy\_s\_3:

**add** [y\_coordinate], **ax**

**call** blackspiece\_1

**sub** [y\_coordinate], **ax**

**pop** **ax**

**ret**

        destroy\_s\_4:

**add** [x\_coordinate], **ax**

**call** blackspiece\_2

**sub** [x\_coordinate], **ax**

**pop** **ax**

**ret**

    destroy\_z:

**push** **ax**

**mov** **ax**, [square\_size]

**cmp** [current\_piece\_rotation], 1

**je** destroy\_z\_1

**cmp** [current\_piece\_rotation], 2

**je** destroy\_z\_2

**cmp** [current\_piece\_rotation], 3

**je** destroy\_z\_3

**cmp** [current\_piece\_rotation], 4

**je** destroy\_z\_4

**ret**

        destroy\_z\_1:

**call** blackzpiece\_1

**pop** **ax**

**ret**

        destroy\_z\_2:

**call** blackzpiece\_2

**pop** **ax**

**ret**

        destroy\_z\_3:

**add** [y\_coordinate], **ax**

**call** blackzpiece\_1

**sub** [y\_coordinate], **ax**

**pop** **ax**

**ret**

        destroy\_z\_4:

**add** [x\_coordinate], **ax**

**call** blackzpiece\_2

**sub** [x\_coordinate], **ax**

**pop** **ax**

**ret**

**endp** destroy\_piece

### proc move\_down\_lines

**תיאור הפרוצדורה**פרוצדורה זו מורידה את כל הריבועים החל משורה מסוימת ריבוע אחד למטה

**משתנים**  
line – השורה ממנה צריך להתחיל למחוק  
x\_coordinate – קואורדינטת ה-x של הריבוע שכעט צריך להוריד  
y\_coordinate – קואורדינטת ה-y של הריבוע שכעט צריך להוריד  
pixelColour – הצבע של הריבוע שצריך להוריד  
main\_colour – הצבע הראשי של הריבוע שיש להוריד  
light\_colour - הצבע הבהיר של הריבוע שיש להוריד  
border\_colour – הצבע הכהה של הריבוע שיש להוריד

**קוד הפרוצדורה**

**proc** move\_down\_lines

**pusha**

**mov** **ax**, [square\_size] ; square size as a register

**mov** **cx**, 10

    move\_down\_lines\_columns:

**push** **cx**

**push** [y\_coordinate]

**mov** **cx**, 21

**sub** **cx**, [line] ; for every line after

    move\_down\_lines\_move\_down\_squares:

**sub** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0

**je** move\_down\_lines\_move\_down\_square\_black

**cmp** [pixelcolour], 0efh

**je** move\_down\_lines\_move\_down\_square\_purple

**cmp** [pixelcolour], 0bfh

**je** move\_down\_lines\_move\_down\_square\_yellow

**cmp** [pixelcolour], 9h

**je** move\_down\_lines\_move\_down\_square\_blue

**cmp** [pixelcolour], 77h

**je** move\_down\_lines\_move\_down\_square\_orange

**cmp** [pixelcolour], 0ffh

**je** move\_down\_lines\_move\_down\_square\_cyan

**cmp** [pixelcolour], 5fh

**je** move\_down\_lines\_move\_down\_square\_red

**cmp** [pixelcolour], 0bdh

**je** move\_down\_lines\_move\_down\_square\_green

**jmp** move\_down\_lines\_move\_down\_squares\_loopend

    move\_down\_lines\_move\_down\_square\_black:

**add** [y\_coordinate], **ax** ; a square down

**mov** [light\_colour], 0 ; black

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**call** drawSquare ; black current square

**sub** [y\_coordinate], **ax** ; return back

**jmp** move\_down\_lines\_move\_down\_squares\_loopEnd

    move\_down\_lines\_move\_down\_square\_yellow:

**mov** [light\_colour], 0 ; black

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**call** drawSquare ; black current square

**add** [y\_coordinate], **ax** ; a square down

**mov** [main\_colour], 37h ; orangish yellow

**mov** [light\_colour], 0bfh ; light yellow

**mov** [border\_colour], 5dh ; brown

**call** drawsquare; redraw it a square down

**sub** [y\_coordinate], **ax** ; return back

**jmp** move\_down\_lines\_move\_down\_squares\_loopEnd

    move\_down\_lines\_move\_down\_square\_purple:

**mov** [light\_colour], 0 ; black

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**call** drawSquare ; black current square

**add** [y\_coordinate], **ax** ; a square down

**mov** [light\_colour], 0efh

**mov** [main\_colour], 0deh

**mov** [border\_colour], 83h

**call** drawsquare; redraw it a square down

**sub** [y\_coordinate], **ax** ; return back

**jmp** move\_down\_lines\_move\_down\_squares\_loopEnd

    move\_down\_lines\_move\_down\_square\_blue:

**mov** [light\_colour], 0 ; black

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**call** drawSquare ; black current square

**add** [y\_coordinate], **ax** ; a square down

**mov** [light\_colour], 9h ; blues

**mov** [main\_colour], 0d0h

**mov** [border\_colour], 40h

**call** drawsquare; redraw it a square down

**sub** [y\_coordinate], **ax** ; return back

**jmp** move\_down\_lines\_move\_down\_squares\_loopEnd

    move\_down\_lines\_move\_down\_square\_orange:

**mov** [light\_colour], 0 ; black

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**call** drawSquare ; black current square

**add** [y\_coordinate], **ax** ; a square down

**mov** [light\_colour], 77h ; orange

**mov** [main\_colour], 27h

**mov** [border\_colour], 15h

**call** drawsquare; redraw it a square down

**sub** [y\_coordinate], **ax** ; return back

**jmp** move\_down\_lines\_move\_down\_squares\_loopEnd

    move\_down\_lines\_move\_down\_square\_cyan:

**mov** [light\_colour], 0 ; black

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**call** drawSquare ; black current square

**add** [y\_coordinate], **ax** ; a square down

**mov** [light\_colour], 0ffh ; cyan

**mov** [main\_colour], 0feh

**mov** [border\_colour], 6h

**call** drawsquare; redraw it a square down

**sub** [y\_coordinate], **ax** ; return back

**jmp** move\_down\_lines\_move\_down\_squares\_loopEnd

    move\_down\_lines\_move\_down\_square\_green:

**mov** [light\_colour], 0 ; black

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**call** drawSquare ; black current square

**add** [y\_coordinate], **ax** ; a square down

**mov** [light\_colour], 0bdh ; greens

**mov** [main\_colour], 38h

**mov** [border\_colour], 22h

**call** drawsquare; redraw it a square down

**sub** [y\_coordinate], **ax** ; return back

**jmp** move\_down\_lines\_move\_down\_squares\_loopEnd

    move\_down\_lines\_move\_down\_square\_red:

**mov** [light\_colour], 0 ; black

**mov** [main\_colour], 0

**mov** [border\_colour], 0

**call** drawSquare ; black current square

**add** [y\_coordinate], **ax** ; a square down

**mov** [light\_colour], 5fh ; reds

**mov** [main\_colour], 0f9h

**mov** [border\_colour], 01h

**call** drawsquare; redraw it a square down

**sub** [y\_coordinate], **ax** ; return back

**jmp** move\_down\_lines\_move\_down\_squares\_loopEnd

    move\_down\_lines\_move\_down\_squares\_loopEnd:

**loop** move\_down\_lines\_move\_down\_squares

**pop** [y\_coordinate]

**pop** **cx**

**add** [x\_coordinate], **ax**

**loop** move\_down\_lines\_columns

**popa**

**ret**

**endp** move\_down\_lines

### proc is\_game\_over

**תיאור הפרוצדורה**פרוצדורה זו בודקת האם יש ריבוע בראש המסך, ואם כן פרוצדורה זו אומרת שהמשחק צריך להיגמר

**משתנים**x\_coordinate – קואורדינטת ה-x של הריבוע שצריך לבדוק  
y\_coordinate – קואורדינטת ה-y של הריבוע שצריך לבדוק  
pixelColour – הצבע הבהיר של הריבוע שנבדק  
game\_over – האם המשחק נגמר

**קוד הפרוצדורה**

**proc** is\_game\_over

**push** [x\_coordinate]

**push** [y\_coordinate]

**push** **ax**

**mov** **ax**, [square\_size]

**mov** [y\_coordinate], 17 ; reset variables

**mov** [x\_coordinate], 144

**mov** **cx**, 4

    is\_game\_over\_loop:

**call** readpixel

**cmp** [pixelcolour], 0

**jne** is\_game\_over\_true

**add** [y\_coordinate], **ax**

**call** readpixel

**cmp** [pixelcolour], 0

**jne** is\_game\_over\_true

**sub** [y\_coordinate], **ax**

**add** [x\_coordinate], **ax**

**loop** is\_game\_over\_loop

**jmp** is\_game\_over\_end

    is\_game\_over\_true:

**mov** [game\_over], 1

    is\_game\_over\_end:

**pop** **ax**

**pop** [y\_coordinate]

**pop** [x\_coordinate]

**ret**

**endp** is\_game\_over

### proc draw\_queue\_thumbnails

**תיאור הפרוצדורה**פרוצדורה זו מציירת את התור

**משתנים**current\_piece – הטטרומינו שיש לצייר בצג התור  
queue – התור  
x\_coordinate – קואורדינטת ה-x של הטטרומינו שצריך לצייר  
y\_coordinate – קואורדינטת ה-y של הטטרומינו שצריך לצייר

**קוד הפרוצדורה**

**proc** draw\_queue\_thumblnails

**pusha**

**push** [current\_piece]

**mov** [x\_coordinate], 250

**mov** [y\_coordinate], 27

**mov** **bx**, **offset** queue

**mov** **si**, 0

**mov** **cx**, 5

    draw\_queue\_thumbnails\_loop:

**push** [**bx+si**]

**pop** [current\_piece]

**mov** [x\_coordinate], 250

**cmp** [current\_piece], 1

**je** draw\_queue\_thumbnail\_io

**cmp** [current\_piece], 4

**je** draw\_queue\_thumbnail\_io

        draw\_queue\_thumbnail:

**call** generate\_piece

**add** [y\_coordinate], 34

**add** **si**, 2

**loop** draw\_queue\_thumbnails\_loop

**jmp** draw\_queue\_thumbnails\_end

        draw\_queue\_thumbnail\_io:

**sub** [x\_coordinate], 4

**jmp** draw\_queue\_thumbnail

    draw\_queue\_thumbnails\_end:

**pop** [current\_piece]

**popa**

**ret**

**endp** draw\_queue\_thumblnails

### proc erase\_queue\_thumbnails

**תיאור הפרוצדורה**פרוצדורה זו מוחקת את התור

**משתנים**current\_piece – הטטרומינו שיש למחוק בצג התור  
queue – התור  
x\_coordinate – קואורדינטת ה-x של הטטרומינו שצריך למחוק  
y\_coordinate – קואורדינטת ה-y של הטטרומינו שצריך למחוק

**קוד הפרוצדורה**

**proc** erase\_queue\_thumblnails

**pusha**

**push** [current\_piece]

**mov** [x\_coordinate], 250

**mov** [y\_coordinate], 27

**mov** **bx**, **offset** queue

**mov** **si**, 0

**mov** **cx**, 5

    erase\_queue\_thumbnails\_loop:

**push** [**bx+si**]

**pop** [current\_piece]

**mov** [current\_piece\_rotation], 1

**mov** [x\_coordinate], 250

**cmp** [current\_piece], 1

**je** erase\_queue\_thumbnail\_io

**cmp** [current\_piece], 4

**je** erase\_queue\_thumbnail\_io

        erase\_queue\_thumbnail:

**call** destroy\_piece

**add** [y\_coordinate], 34

**add** **si**, 2

**loop** erase\_queue\_thumbnails\_loop

**jmp** erase\_queue\_thumbnails\_end

        erase\_queue\_thumbnail\_io:

**sub** [x\_coordinate], 4

**jmp** erase\_queue\_thumbnail

    erase\_queue\_thumbnails\_end:

**pop** [current\_piece]

**popa**

**ret**

**endp** erase\_queue\_thumblnails

### proc draw\_held\_piece\_thumbnails

**תיאור הפרוצדורה**פרוצדורה זו מציירת את הטטרומינו בתא השמירה

**משתנים**current\_piece – הטטרומינו שיש לצייר בתא השמירה  
held\_piece – הטטרומינו השמור

**קוד הפרוצדורה**

**proc** draw\_held\_piece\_thumbnail

**pusha**

**push** [current\_piece]

**mov** [y\_coordinate], 38

**push** [held\_piece]

**pop** [current\_piece]

**cmp** [current\_piece], 1

**je** draw\_held\_piece\_thumbnail\_io

**cmp** [current\_piece], 4

**je** draw\_held\_piece\_thumbnail\_io

**mov** [x\_coordinate], 46

**call** generate\_piece

**pop** [current\_piece]

**popa**

**ret**

        draw\_held\_piece\_thumbnail\_io:

**mov** [x\_coordinate], 42

**call** generate\_piece

**pop** [current\_piece]

**popa**

**ret**

**endp** draw\_held\_piece\_thumbnail

### proc erase\_held\_piece\_thumbnails

**תיאור הפרוצדורה**פרוצדורה זו מוחקת את הטטרומינו בתא השמירה

**משתנים**current\_piece – הטטרומינו שיש לצייר בתא השמירה  
held\_piece – הטטרומינו השמור

**קוד הפרוצדורה**

**proc** black\_held\_piece\_thumbnail

**pusha**

**push** [current\_piece]

**mov** [y\_coordinate], 38

**push** [held\_piece]

**pop** [current\_piece]

**cmp** [current\_piece], 1

**je** black\_held\_piece\_thumbnail\_io

**cmp** [current\_piece], 4

**je** black\_held\_piece\_thumbnail\_io

**mov** [x\_coordinate], 46 ; regular thumbnail position

**mov** [current\_piece\_rotation], 1

**call** destroy\_piece

**pop** [current\_piece]

**popa**

**ret**

        black\_held\_piece\_thumbnail\_io: ; i and o thumbnail position

**mov** [x\_coordinate], 42

**mov** [current\_piece\_rotation], 1

**call** destroy\_piece

**pop** [current\_piece]

**popa**

**ret**

**endp** black\_held\_piece\_thumbnail

### proc draw\_score

**תיאור הפרוצדורה**פרוצדורה זו מדפיסה את הניקוד

**משתנים**  
score – הניקוד

**קוד הפרוצדורה**

**proc** draw\_score

**pusha**

**mov** **bx**, **offset** score

**mov** **dx**, **offset** score

**mov** **si**, 0

**mov** **cx**, 10

    draw\_score\_add\_loop:

**mov** **al**, '0'

**add** [**bx+si**], **al**

**inc** **si**

**loop** draw\_score\_add\_loop

**call** print\_text

**mov** **si**, 0

**mov** **cx**, 10

    draw\_score\_sub\_loop:

**mov** **al**, '0'

**sub** [**bx+si**], **al**

**inc** **si**

**loop** draw\_score\_sub\_loop

**popa**

**ret**

**endp** draw\_score

### פרוצדורות הגדלת הניקוד

**תיאור הפרוצדורות**  
פרוצדורות אלו מגדילות את הניקוד

**משתנים**score – הניקוד

**קוד הפרוצדורות**

**proc** inc\_score\_first\_digit

**pusha**

**mov** **bx**, **offset** score

**mov** **si**, 9

**mov** **cx**, 10

    inc\_digit\_1:

**inc** [**bx+si**]

**mov** **dl**, 9

**cmp** [**bx+si**], **dl**

**ja** digit\_overflow\_1

**popa**

**ret**

    digit\_overflow\_1:

**mov** **dl**, 0

**mov** [**bx+si**], **dl**

**dec** **si**

**loop** inc\_digit\_1

**popa**

**ret**

**endp** inc\_score\_first\_digit

**proc** inc\_score\_second\_digit

**pusha**

**mov** **bx**, **offset** score

**mov** **si**, 8

**mov** **cx**, 9

    inc\_digit\_2:

**inc** [**bx+si**]

**mov** **dl**, 9

**cmp** [**bx+si**], **dl**

**ja** digit\_overflow\_2

**popa**

**ret**

    digit\_overflow\_2:

**mov** **dl**, 0

**mov** [**bx+si**], **dl**

**dec** **si**

**loop** inc\_digit\_2

**popa**

**ret**

**endp** inc\_score\_second\_digit

**proc** inc\_score\_third\_digit

**pusha**

**mov** **bx**, **offset** score

**mov** **si**, 7

**mov** **cx**, 8

    inc\_digit\_3:

**inc** [**bx+si**]

**mov** **dl**, 9

**cmp** [**bx+si**], **dl**

**ja** digit\_overflow\_3

**popa**

**ret**

    digit\_overflow\_3:

**mov** **dl**, 0

**mov** [**bx+si**], **dl**

**dec** **si**

**loop** inc\_digit\_3

**popa**

**ret**

**endp** inc\_score\_third\_digit

### proc draw\_level

**תיאור הפרוצדורה**פרוצדורה זו מדפיסה את השלב הנוכחי

**משתנים**level – השלב

**קוד הפרוצדורה**

**proc** draw\_level

**pusha**

**mov** **bx**, **offset** level

**mov** **dx**, **offset** level

**mov** **si**, 0

**mov** **cx**, 2

    draw\_level\_add\_loop:

**mov** **al**, '0'

**add** [**bx+si**], **al**

**inc** **si**

**loop** draw\_level\_add\_loop

**call** print\_text

**mov** **si**, 0

**mov** **cx**, 2

    draw\_level\_sub\_loop:

**mov** **al**, '0'

**sub** [**bx+si**], **al**

**inc** **si**

**loop** draw\_level\_sub\_loop

**popa**

**ret**

**endp** draw\_level

### proc calculate\_level

**תיאור הפרוצדורה**פרוצדורה זו מחשבת את השלב הנוכחי

**משתנים**lines\_cleared – כמות השורות שנוקו, לפי מספר זה מחשבים את השלה  
level – השלב הנוכחי  
level\_num – מספר השלב הנוכחי, בצורה שנוח לבצע חישובים איתה  
default\_speed – המהירות של הנפילה, תלויה בשלב

**קוד הפרוצדורה**

**proc** calculate\_level

**pusha**

**mov** **bx**, **offset** level

**cmp** [lines\_cleared], 5

**jb** level\_0

**cmp** [lines\_cleared], 10

**jb** level\_1

**cmp** [lines\_cleared], 15

**jb** level\_2

**cmp** [lines\_cleared], 20

**jb** level\_3

**cmp** [lines\_cleared], 30

**jb** level\_4

**cmp** [lines\_cleared], 40

**jb** level\_5

**cmp** [lines\_cleared], 50

**jb** level\_6

**cmp** [lines\_cleared], 60

**jb** level\_7

**cmp** [lines\_cleared], 70

**jb** level\_8

**cmp** [lines\_cleared], 80

**jb** level\_9

**cmp** [lines\_cleared], 100

**jb** level\_10

**cmp** [lines\_cleared], 120

**jb** level\_11

**cmp** [lines\_cleared], 140

**jb** level\_12

**cmp** [lines\_cleared], 160

**jb** level\_13

**cmp** [lines\_cleared], 180

**jb** level\_14

**cmp** [lines\_cleared], 200

**jb** level\_15

**cmp** [lines\_cleared], 240

**jb** level\_16

**cmp** [lines\_cleared], 280

**jb** level\_17

**cmp** [lines\_cleared], 330

**jb** level\_18

**cmp** [lines\_cleared], 400

**jb** level\_19

**cmp** [lines\_cleared], 500

**jb** level\_20

**cmp** [lines\_cleared], 600

**jb** level\_21

**cmp** [lines\_cleared], 700

**jb** level\_22

**cmp** [lines\_cleared], 800

**jb** level\_23

**cmp** [lines\_cleared], 950

**jb** level\_24

**jmp** level\_25

    level\_0:

**mov** [level\_num], 0

**mov** **al**, 0

**mov** [**bx+**1], **al**

**mov** **al**, 0

**mov** [**bx+**0], **al**

**mov** [default\_speed], 7FFFh

**popa**

**ret**

    level\_1:

**mov** [level\_num], 1

**mov** **al**, 1

**mov** [**bx+**1], **al**

**mov** **al**, 0

**mov** [**bx+**0], **al**

**mov** [default\_speed], 7000h

**popa**

**ret**

    level\_2:

**mov** [level\_num], 2

**mov** **al**, 2

**mov** [**bx+**1], **al**

**mov** **al**, 0

**mov** [**bx+**0], **al**

**mov** [default\_speed], 6800h

**popa**

**ret**

    level\_3:

**mov** [level\_num], 3

**mov** **al**, 3

**mov** [**bx+**1], **al**

**mov** **al**, 0

**mov** [**bx+**0], **al**

**mov** [default\_speed], 6000h

**popa**

**ret**

    level\_4:

**mov** [level\_num], 4

**mov** **al**, 4

**mov** [**bx+**1], **al**

**mov** **al**, 0

**mov** [**bx+**0], **al**

**mov** [default\_speed], 5800h

**popa**

**ret**

    level\_5:

**mov** [level\_num], 5

**mov** **al**, 5

**mov** [**bx+**1], **al**

**mov** **al**, 0

**mov** [**bx+**0], **al**

**mov** [default\_speed], 4800h

**popa**

**ret**

    level\_6:

**mov** [level\_num], 6

**mov** **al**, 6

**mov** [**bx+**1], **al**

**mov** **al**, 0

**mov** [**bx+**0], **al**

**mov** [default\_speed], 4000h

**popa**

**ret**

    level\_7:

**mov** [level\_num], 7

**mov** **al**, 7

**mov** [**bx+**1], **al**

**mov** **al**, 0

**mov** [**bx+**0], **al**

**mov** [default\_speed], 3800h

**popa**

**ret**

    level\_8:

**mov** [level\_num], 8

**mov** **al**, 8

**mov** [**bx+**1], **al**

**mov** **al**, 0

**mov** [**bx+**0], **al**

**mov** [default\_speed], 3000h

**popa**

**ret**

    level\_9:

**mov** [level\_num], 9

**mov** **al**, 9

**mov** [**bx+**1], **al**

**mov** **al**, 0

**mov** [**bx+**0], **al**

**mov** [default\_speed], 2800h

**popa**

**ret**

    level\_10:

**mov** [level\_num], 10

**mov** **al**, 0

**mov** [**bx+**1], **al**

**mov** **al**, 1

**mov** [**bx+**0], **al**

**mov** [default\_speed], 2000h

**popa**

**ret**

    level\_11:

**mov** [level\_num], 11

**mov** **al**, 1

**mov** [**bx+**1], **al**

**mov** **al**, 1

**mov** [**bx+**0], **al**

**mov** [default\_speed], 1A00h

**popa**

**ret**

    level\_12:

**mov** [level\_num], 12

**mov** **al**, 2

**mov** [**bx+**1], **al**

**mov** **al**, 1

**mov** [**bx+**0], **al**

**mov** [default\_speed], 1400h

**popa**

**ret**

    level\_13:

**mov** [level\_num], 13

**mov** **al**, 3

**mov** [**bx+**1], **al**

**mov** **al**, 1

**mov** [**bx+**0], **al**

**mov** [default\_speed], 1000h

**popa**

**ret**

    level\_14:

**mov** [level\_num], 14

**mov** **al**, 4

**mov** [**bx+**1], **al**

**mov** **al**, 1

**mov** [**bx+**0], **al**

**mov** [default\_speed], 0b00h

**popa**

**ret**

    level\_15:

**mov** [level\_num], 15

**mov** **al**, 5

**mov** [**bx+**1], **al**

**mov** **al**, 1

**mov** [**bx+**0], **al**

**mov** [default\_speed], 800h

**popa**

**ret**

    level\_16:

**mov** [level\_num], 16

**mov** **al**, 6

**mov** [**bx+**1], **al**

**mov** **al**, 1

**mov** [**bx+**0], **al**

**mov** [default\_speed], 600h

**popa**

**ret**

    level\_17:

**mov** [level\_num], 17

**mov** **al**, 7

**mov** [**bx+**1], **al**

**mov** **al**, 1

**mov** [**bx+**0], **al**

**mov** [default\_speed], 500h

**popa**

**ret**

    level\_18:

**mov** [level\_num], 18

**mov** **al**, 8

**mov** [**bx+**1], **al**

**mov** **al**, 1

**mov** [**bx+**0], **al**

**mov** [default\_speed], 400h

**popa**

**ret**

    level\_19:

**mov** [level\_num], 19

**mov** **al**, 9

**mov** [**bx+**1], **al**

**mov** **al**, 1

**mov** [**bx+**0], **al**

**mov** [default\_speed], 375h

**popa**

**ret**

    level\_20:

**mov** [level\_num], 20

**mov** **al**, 0

**mov** [**bx+**1], **al**

**mov** **al**, 2

**mov** [**bx+**0], **al**

**mov** [default\_speed], 200h

**popa**

**ret**

    level\_21:

**mov** [level\_num], 21

**mov** **al**, 1

**mov** [**bx+**1], **al**

**mov** **al**, 2

**mov** [**bx+**0], **al**

**mov** [default\_speed], 100h

**popa**

**ret**

    level\_22:

**mov** [level\_num], 22

**mov** **al**, 2

**mov** [**bx+**1], **al**

**mov** **al**, 2

**mov** [**bx+**0], **al**

**mov** [default\_speed], 50h

**popa**

    level\_23:

**mov** [level\_num], 23

**mov** **al**, 3

**mov** [**bx+**1], **al**

**mov** **al**, 2

**mov** [**bx+**0], **al**

**mov** [default\_speed], 10h

**popa**

    level\_24:

**mov** [level\_num], 24

**mov** **al**, 4

**mov** [**bx+**1], **al**

**mov** **al**, 2

**mov** [**bx+**0], **al**

**mov** [default\_speed], 1h

**popa**

    level\_25:

**mov** [level\_num], 25

**mov** **al**, 5

**mov** [**bx+**1], **al**

**mov** **al**, 2

**mov** [**bx+**0], **al**

**mov** [default\_speed], 0h

**popa**

**ret**

**endp** calculate\_level

### proc draw\_cleared\_lines

**תיאור הפרוצדורה**פרוצדורה זו מדפיסה את כמות השורות שהשחקן ניקה

**משתנים**lines\_cleared\_printable – כמות השורות שנוקו בצורה שקל להדפיס

**קוד הפרוצדורה**

**proc** draw\_cleared\_lines

**pusha**

**mov** **bx**, **offset** lines\_cleared\_printable

**mov** **dx**, **offset** lines\_cleared\_printable

**mov** **si**, 0

**mov** **cx**, 3

    draw\_cleared\_lines\_add\_loop:

**mov** **al**, '0'

**add** [**bx+si**], **al**

**inc** **si**

**loop** draw\_cleared\_lines\_add\_loop

**call** print\_text

**mov** **si**, 0

**mov** **cx**, 3

    draw\_cleared\_lines\_sub\_loop:

**mov** **al**, '0'

**sub** [**bx+si**], **al**

**inc** **si**

**loop** draw\_cleared\_lines\_sub\_loop

**popa**

**ret**

**endp** draw\_cleared\_lines

### proc inc\_cleared\_lines

**תיאור הפרוצדורה**פרוצדורה זו מגדילה את הערך בכמות השורות שנוקו

**משתנים**lines\_cleared\_printable – כמות השורות שנוקו בצורה שקל להדפיס

**קוד הפרוצדורה**

**proc** inc\_cleared\_lines

**pusha**

**mov** **bx**, **offset** lines\_cleared\_printable

**mov** **si**, 2

**mov** **cx**, 3

    inc\_digit\_cleared\_lines:

**inc** [**bx+si**]

**mov** **dl**, 9

**cmp** [**bx+si**], **dl**

**ja** digit\_overflow\_cleared\_lines

**popa**

**ret**

    digit\_overflow\_cleared\_lines:

**mov** **dl**, 0

**mov** [**bx+si**], **dl**

**dec** **si**

**loop** inc\_digit\_cleared\_lines

**popa**

**ret**

**endp** inc\_cleared\_lines

# אלגוריתם

## מסך הפתיחה

**call** entergraphicmode

**mov** **cx**, **offset** filename2 ; print screen

**mov** [filename], **cx**

**call** OPENBITMAP

זימון פעולה הנכנסת למצב גרפי

השמה של כתובת המשתנה filename2 למשתנה filename

זימון פעולה אשר מדפיסה את קובץ הBMP ששמו נמצא בכתובת אשר רשומה המשתנה filename

**cmp** [pressedkey], '1'

**je** level\_1\_start

**cmp** [pressedkey], '2'

**je** level\_2\_start

**cmp** [pressedkey], '3'

**je** level\_3\_start

**cmp** [pressedkey], '4'

**je** level\_4\_start

**cmp** [pressedkey], '5'

**je** level\_5\_start

**cmp** [pressedkey], '6'

**je** level\_6\_start

**cmp** [pressedkey], '7'

**je** level\_7\_start

**cmp** [pressedkey], '8'

**je** level\_8\_start

**cmp** [pressedkey], '9'

**je** level\_9\_start

**jmp** game\_start

אם המקש שנלחץ הוא 1

אם כן, קפיצה לתווית level\_1\_start

אם המקש שנלחץ הוא 2

אם כן, קפיצה לתווית level\_2\_start

אם המקש שנלחץ הוא 3

אם כן, קפיצה לתווית level\_3\_start

אם המקש שנלחץ הוא 4

אם כן, קפיצה לתווית level\_4\_start

אם המקש שנלחץ הוא 5

אם כן, קפיצה לתווית level\_5\_start

אם המקש שנלחץ הוא 6

אם כן, קפיצה לתווית level\_6\_start

אם המקש שנלחץ הוא 7

אם כן, קפיצה לתווית level\_7\_start

אם המקש שנלחץ הוא 8

אם כן, קפיצה לתווית level\_8\_start

אם המקש שנלחץ הוא 9

אם כן, קפיצה לתווית level\_9\_start

קופץ לתווית game\_start

    level\_1\_start:

**mov** [lines\_cleared], 5

**jmp** game\_start

    level\_2\_start:

**mov** [lines\_cleared], 10

**jmp** game\_start

    level\_3\_start:

**mov** [lines\_cleared], 15

**jmp** game\_start

    level\_4\_start:

**mov** [lines\_cleared], 20

**jmp** game\_start

    level\_5\_start:

**mov** [lines\_cleared], 30

**jmp** game\_start

    level\_6\_start:

**mov** [lines\_cleared], 40

**jmp** game\_start

    level\_7\_start:

**mov** [lines\_cleared], 50

**jmp** game\_start

    level\_8\_start:

**mov** [lines\_cleared], 60

**jmp** game\_start

    level\_9\_start:

**mov** [lines\_cleared], 70

**jmp** game\_start

**תווית – level\_1\_start** השמה 5 למשתנה lines\_cleared  
 קפיצה לתווית game\_start

**תווית – level\_2\_start** השמה 10 למשתנה lines\_cleared  
 קפיצה לתווית game\_start

**תווית – level\_3\_start** השמה 15 למשתנה lines\_cleared  
 קפיצה לתווית game\_start

**תווית – level\_4\_start** השמה 20 למשתנה lines\_cleared  
 קפיצה לתווית game\_start

**תווית – level\_5\_start** השמה 30 למשתנה lines\_cleared  
 קפיצה לתווית game\_start

**תווית – level\_6\_start** השמה 40 למשתנה lines\_cleared  
 קפיצה לתווית game\_start

**תווית – level\_7\_start** השמה 50 למשתנה lines\_cleared  
 קפיצה לתווית game\_start

**תווית – level\_8\_start** השמה 60 למשתנה lines\_cleared  
 קפיצה לתווית game\_start

**תווית – level\_9\_start** השמה 70 למשתנה lines\_cleared  
 קפיצה לתווית game\_start

## המשחק עצמו

### איתחול ערכים

game\_start:

**call** initializerandom

    ; initialize queue

**call** generate\_last\_7\_queue

**mov** **bx**, **offset** queue

**mov** **cx**, 7

    initial\_move\_queue\_7\_spots:

**push** **cx**

**mov** **si**, 2

**mov** **cx**, 13

        initial\_move\_queue\_loop:

**push** [**bx+si**]

**sub** **si**, 2

**pop** [**bx+si**]

**add** **si**, 4

**loop** initial\_move\_queue\_loop

**pop** **cx**

**loop** initial\_move\_queue\_7\_spots

**call** generate\_last\_7\_queue

**mov** [queue\_iteration], 0

**תווית – game\_start** זימון פעולה המאתחלת את זרע הראנדום

זימון פעולה המאתחלת את 7 המקומות האחרונים

השמה המקום של מערך queue לרגיסטר bx

השמה 7 לרגיסטר cx

**תווית – initial\_move\_queue\_7\_spots** דחיפת רגיסטר cx למחסנית

השמה 2 לרגיסטר si

השמה 13 לרגיסטר cx

**תווית - initial\_move\_queue\_loop** השמת הערך במקום הbx+si-2 לערך במקום הbx+si-2

הוספת 2 לרגיסטר si

לופ לתווית initial\_move\_queue\_loop

משיכה מהמחסנית לרגיסטר cx

לופ לתווית initial\_move\_queue\_7\_spots

    ; Process BMP file

**call** entergraphicmode

**mov** **cx**, **offset** filename1

**mov** [filename], **cx**

**call** openbitmap

**push** 3 ;x coordinate

**push** 15 ;y coordinate

**call** cursor\_location

**call** draw\_score

**push** 11 ;x coordinate

**push** 13 ;y coordinate

**call** cursor\_location

**call** draw\_level

**push** 10 ;x coordinate

**push** 17 ;y coordinate

**call** cursor\_location

**call** draw\_cleared\_lines

זימון פעולה אשר מכניסה למצב גרפי

השמת כתובת המשתנה filename1 למשתנה filename

זימון פעולה אשר מדפיסה את הקובץ ששמו נמצא בכתובת אשר נמצאת במשתנה filename

זימון פרוצדורה אשר מניחה את סמן הטקסט בx=3,y=5

זימון פרוצדורה אשר מדפיסה את הניקוד

זימון פרוצדורה אשר מניחה את סמן הטקסט בx=11,y=13

זימון פרוצדורה אשר מדפיסה את השלב הנוכחי

זימון פרוצדורה אשר מניחה את סמן הטקסט בx=10,y=17

זימון פרוצדורה אשר מדפיסה את כמות השורות שנוקו

### מהלך המשחק

mainGameLoop:

    ; reset hard variables (so mechanisms like hold won't reset them)

**mov** [held\_this\_turn], 0

**mov** [lines\_cleared\_this\_turn], 0

**תווית - mainGameLoop**

השמה 0 למשתנה held\_this\_turn

השמה 0 למשתנה lines\_cleared\_this\_turn

; this code segment checks each row and if every square in it isn't empty (not black) if it is, this segment empties the row

**mov** [x\_coordinate], 120

**mov** [y\_coordinate], 17

**mov** **ax**, [square\_size] ; square size as a register

**mov** **cx**, 21 ; for each row

    clearing\_row\_mechanism:

**push** **cx**

**mov** **cx**, 10 ; for every column in this row

        check\_row\_for\_full:

**call** readpixel

**cmp** [pixelcolour], 0 ; check each square if it's empty

**je** finished\_clearing\_row\_mechanism ; if a square is empty, finish without clearing the row and move on to the next one

**add** [x\_coordinate], **ax**

**loop** check\_row\_for\_full

        ; reaches here only if the whole row isn't empty

**mov** [x\_coordinate], 120 ; reset x coord

**mov** [main\_colour], 0 ; black

**mov** [light\_colour], 0

**mov** [border\_colour], 0

**mov** **cx**, 10 ; for 10 squares

        empty\_row\_columns:

**call** drawsquare ; clear the square

**add** [x\_coordinate], **ax**

**loop** empty\_row\_columns

**pop** [line] ; get the line number to line from the stack

**push** [line]

**mov** [x\_coordinate], 120 ; reset x coord

**call** move\_down\_lines

**inc** [lines\_cleared\_this\_turn]

**inc** [lines\_cleared]

**call** inc\_cleared\_lines

    finished\_clearing\_row\_mechanism:

**mov** [x\_coordinate], 120 ; reset x coord

**add** [y\_coordinate], **ax** ; next row

**pop** **cx**

**loop** clearing\_row\_mechanism

השמה 120 למשתנה x\_coordinate

השמה 17 למשתנה y\_coordinate

השמה 20 לרגיסטר cx

**תווית – clearing\_row\_mechanism**

דחיפת רגיסטר cx למחסנית

השמה 10 לרגיסטר cx

**תווית – check\_row\_for\_full**

זימון פעולה אשר קוראת את צבע הפיקסל הנוכחי

משווה את צבע הריבוע ל0 (שחור)

אם שווה, קפיצה לתווית finished\_clearing\_row\_mechanism

הוספת משתנה square\_size

לופ לתווית check\_row\_for\_full

השמה 120 למשתנה x\_coordinate

השמה 0 (שחור) למשתנים main\_colour, light\_colour, border\_colour

השמה 10 לרגיסטר cx

**תווית – empty row columns**

זימון פעולה אשר מציירת ריבוע

הוספת ערך המשתנה square\_size למשתנה x\_coordinate

לופ לתווית empty row columns

משיכה מהמחסנית למשתנה lines

דחיפת ערך המשתנה lines למחסנית

השמה 120 למשתנה x\_coordinate

זימון פרוצדורה שמורידה את כל השורות מעל השורה במשתנה line

הגדלת הערך במשתנים lines\_cleared\_this\_turn,lines\_cleared ב-1

זימון פרוצדורה אשר מוסיפה 1 למשתנה lines\_cleared\_printable

**תווית – finished\_clearing\_row\_mechanism**

השמת ערך משתנה square\_size למשתנה y\_coordinate

הוספת ערך משתנה square\_size למשתנה y\_coordinate

משיכה מהמכסנית לרגיסטר cx

לופ לתווית clearing\_row\_mechanism