**שם הפרויקט: MIRI’S GAME**

**שם פרטי: מירי**

**שם משפחה: וולוז'ינסקי**

**שם בית הספר: גמנסיה העברית הרצליה**

**שם המורה: שלומי אח-נין**

**סביבת עבודה: emu8086**

**תאריך הגשה: 20.3.2015**



**תוכן עניינים:**

**מבוא והוראות**...................................................3

**הוראות לבוחן**...................................................4

**קשיים**................................................................5

**תיאור המשתנים**...............................................6

**תיאור הפרוצדורות**...........................................7

**תרשים זרימה**...................................................28

**תמונות מסך של המשחק**.................................34 **קוד**....................................................................36

**סיכום ותודות**....................................................59

**ביוגרפיה**...........................................................60

**מבוא והוראות:**

בחרתי לבנות מבוך משום שמבוך זהו משחק שדי שיחקתי בו הרבה כשהייתי קטנה.

חשבתי כי מבוך הוא משחק מאוד פשוט וכי כל כיתתי תעשה אותו, לכן הוספתי עלילה, מלא צבעים ואפילו חתונה! על מנת להפוך משחק זה למשחק צבעוני ונחמד.  
משחק זה הוא מבוך יחסית קל ופשוט עם עלילה.

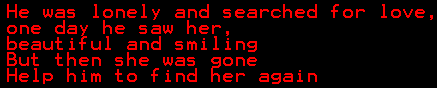
המשחק כולל מלא צבעים ובעל 1300-1200 שורות.

כתבתי אותו חודש- חודשיים ונעזרתי מלא באינטרנט ומשם קיבלתי רעיונות על עיצוב המשחק.

מטרת המשחק היא להביא את הסמיילי הכחול לסמיילי הצהוב.   
ההוראות הם: 8 כדי לזוז למעלה, 2 למטה , 6 ימינה ו4 שמאלה.  
כמו כן ניתן לזוז באלכסון: 9 אלכסון למעלה ימינה, 3 אלכסון למטה ימינה, 7 אלכסון למעלה שמאלה ו1 אלכסון למטה שמאלה.



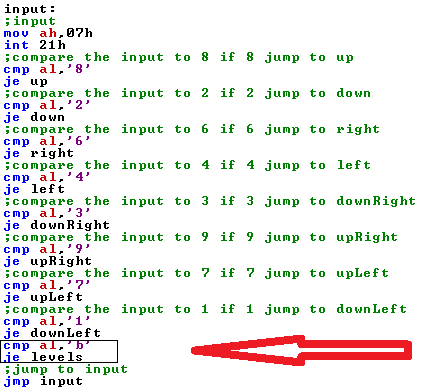
העלילה היא:  
הסמיילי הכחול חיפש אהבה והוא היה בודד, יום אחד מצא סמיילי יפה ומחייך שהוא התאהב בו אבל אז הוא איבד אותו. תעזרו לו למצוא אותו.



**הוראות לבוחן**:

לבוחן היקר, ראשית אני מקווה שתיתן לי את הציון המגיע לי ושתהנה מבדיקת עבודתי.

על מנת לחסוך בזמנך ולא לסיים כל שלב כדי לעבור אותו, הנך יכול להדפיס זאת ב Input:



כשתלחץ b באחד השלבים (לא כולל השלב האחרון- החתונה) תעבור שלב מבלי לסיימו.

את המשחק יש להריץ ב8086EMU.

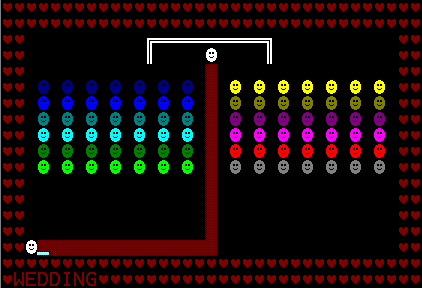
בדיקה נעימה,

מירי

**קשיים:**

הקושי המשמעותי ביותר שנתקלתי בו הוא העיצוב של המשחק. המשחק מלא בצבעים והתקשיתי לצבוע פריטים קטנים.

בשלב האחרון שזהו החתונה הייתי צריכה להדפיס כל שורה מחלק כלשהו בחתונה שרציתי לצבוע בנפרד וזה לקח המון עבודה וחישובים.



חוץ מבעיות העיצוב לא נתקלתי בקשיים משמעותיים.

**תיאור המשתנים:**

מיקומו של השחקן בשלושת השלבים הראשונים:  
x1- משתנה המייצג את העמודה שבה נמצא השחקן   
y1- משתנה המייצג את השורה שבה נמצא השחקן  
x2- משתנה המייצג את העמודה הקודמת שהיה בה השחקן  
y2- משתנה המייצג את השורה הקודמת שהיה בה השחקן

Level- משתנה המייצג את השלב

מיקומו של הכלה בשלב החתונה:  
x1F- משתנה המייצג את העמודה שבה נמצאת הכלה  
y1F- משתנה המייצג את השורה שבה נמצאת הכלה  
x2F- משתנה המייצג את העמודה הקודמת שהייתה בה הכלה  
y2F- משתנה המייצג את השורה הקודמת שהייתה בה הכלה

מחרוזות:

Del- משתנה המייצג מחרוזת שמודפסת לאחר תנועת הסמיילי

delF- משתנה המייצג מחרוזת המודפסת בשלב החתונה לאחר תזוזת הסמיילי

Play- משתנה המייצג מחרוזת של השחקן

Girlfriend- משתנה המייצג מחרוזת של המטרה של השחקן (לאן השחקן צריך להגיע)

Miri- משתנה המייצג מחרוזת של מסך ההפעלה

Story- משתנה המייצג מחרוזת של עלילת המשחק

Ins- משתנה המייצג מחרוזת של ההוראות המשחק

Sad- משתנה המייצג מחרוזת של הפסד המשחק

familyTree- משתנה המייצג מחרוזת של סיום המשחק

maze1- משתנה המייצג מחרוזת של השלב הראשון

maze2- משתנה המייצג מחרוזת של השלב השני

maze3- משתנה המייצג מחרוזת של השלב השלישי

Clear- משתנה המייצג מחרוזת של לוח המחיקה

Weding- משתנה המייצג מחרוזת של השלב החתונה

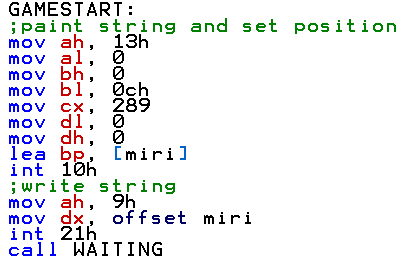
wed1- משתנה המייצג את צלעות החופה בחתונה

wed2- משתנה המייצג את תקרת החופה בחתונה

Families- משתנה המייצג את קבוצות הסמיילים בחתונה

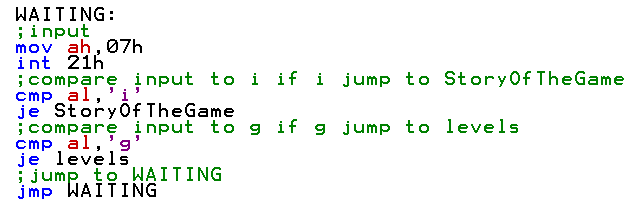
**תיאור הפרוצדורות:**

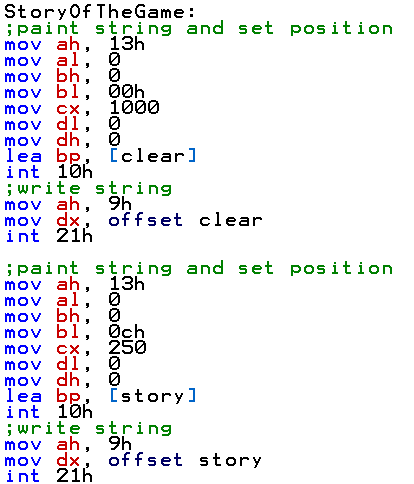
**פרוצדורת GAMESTART:**פרוצדורה זו מדפיסה את miri וצובעת אותה באדום בהיר רק 289 פיקסלים ממנה.



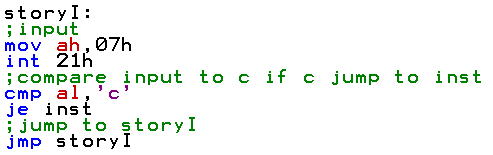
**פרוצדורת WAITING:**פרוצדורה זו קולטת תו  
אם התו הוא I היא תקפוץ לפרוצדורה StoryOfTheGame

אם G היא תקפוץ לפרוצדורה Levels

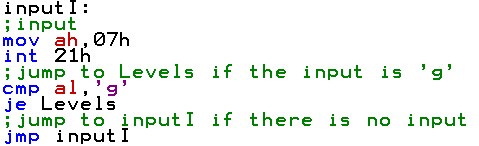




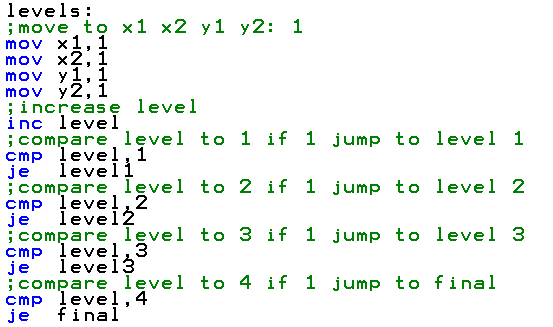
**פרוצדורת StoryOfTheGame:**פרוצדורה זו מוחקת את כל מה שיש במסך.  
ומדפיסה את Story בצבע אדום בהיר



**פרוצדורת storyI:**פרוצדורה זו קולטת תו.  
אם התו הוא C היא קופצת לinst

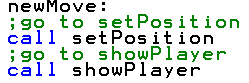


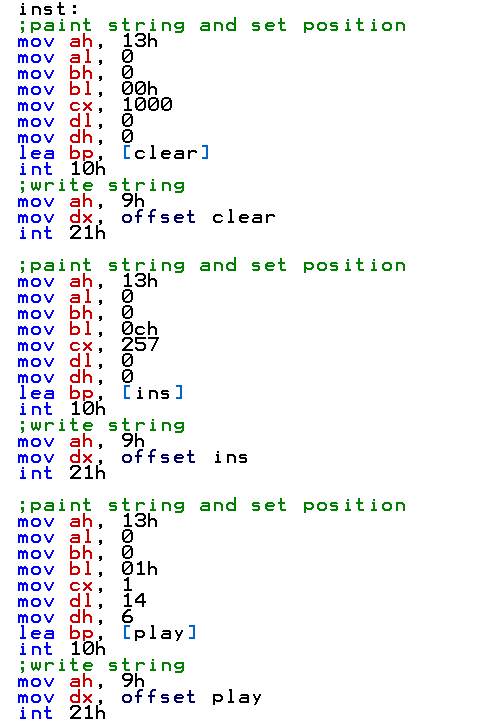
**פרוצדורת inputI :**פרוצדורה זו קולטת תו ואם הוא G היא קופצת לLevels



**פרוצדורת Levels :**פרוצדורה זו מגדילה את המשתנים באחד ומכניסה למיקום של השחקן 1 בכל ערכיו.  
היא בודקת מה הערך של המשתנה level וקופצת בהתאם לשלב המתאים לו.

**פרוצדורת newMove:**פרוצדורה זו קוראת לפרוצדורות:  
setPosition  
shopPlayer

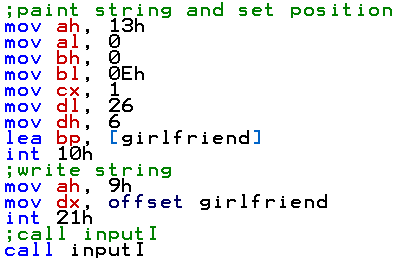


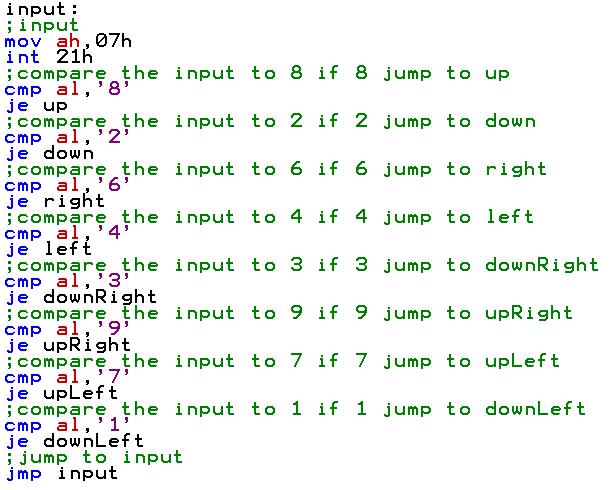


**פרוצדורת inst:**פרוצדורה זו מוחקת את המסך ומדפיסה עליו במקומות שונים את:

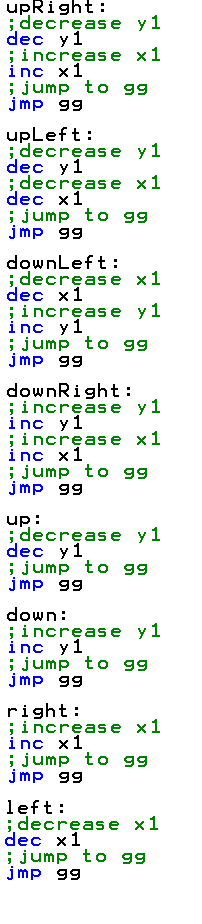
ins , play,girlfriend

וצובעת אותם





**פרוצדורת input:**פרוצדורה זו קולטת תווים וקופצת בהתאם לערכם



**פרוצדורת:**upright- מזיזה את השחקן באלכסון ימין למעלה  
upLeft- מזיזה את השחקן באלסכון שמאל למעלה

downLeft- מזיזה את השחקן באלכסון שמאל למטה

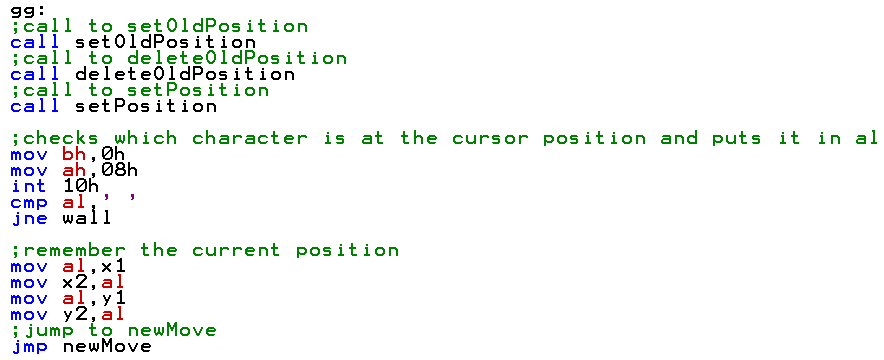
downRight- מזיזה את השחקן באלכסון ימין למטה

Up- מזיזה את השחקן למעלה

down מזיזה את השחקן למטה

Right- מזיזה את השחקן ימינה

Left- מזיזה את השחקן שמאלה



**פרוצדורת gg:**

קורא ל:

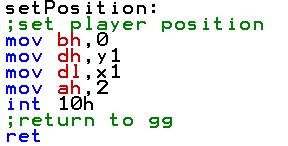
setOldPosition

deleteOldPosition

setPosition

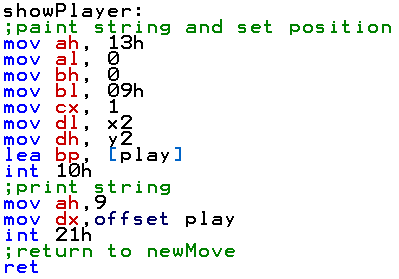
ובודק איפה נמצא השחקן, אם השחקן במקום ריק הוא קופץ לwall

והפרוצדורה מכניסה את הערך שבX1 לX2 ושל Y1 לY2  
בסוף הפרוצדורה היא קופצת לnewMove



**פרוצדורת setPosition:**

הפרוצדורה מגדירה את המיקום של השחקן

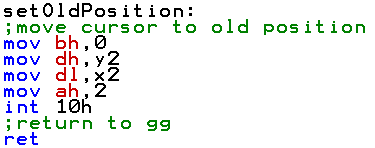


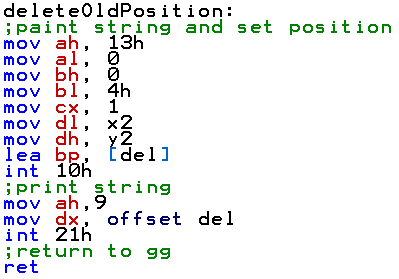
**פרוצדורת setlOldPosition:**

הפרוצדורה מגדירה את המיקום הישן של השחקן

**פרוצדורת showPlayer:**

הפרוצדורה מדפיסה את השחקן במיקום של X1 Y1 וצובעת אותו בכחול





**פרוצדורת wall:**

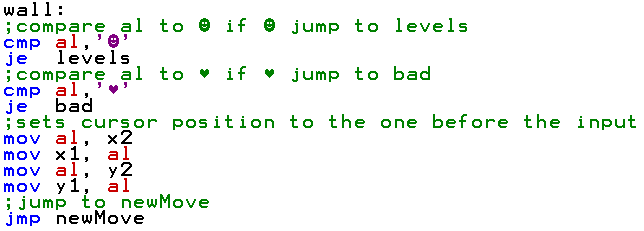
בודק איפה נמצא השחקן  
אם השחקן נמצא על סמיילי, יקפוץ לlevels

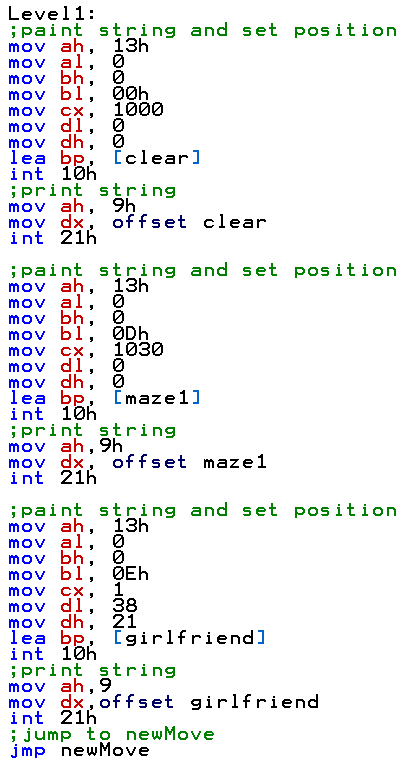
אם השחקן נמצא על לב, יקפוץ ל bad

הפרוצדורה שם את הערך של X2 בX1 ושל Y2 בY1

**פרוצדורת deleteOldPosition:**

מדפיסה במיקום הקודם של השחקן את del

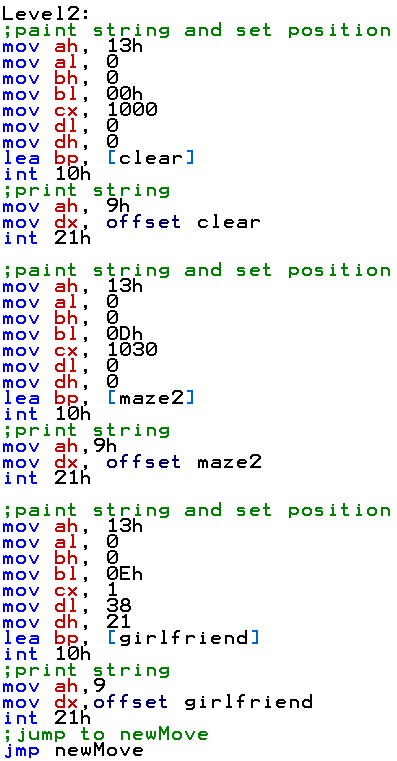




**פרוצדורת Level1:**

הפרוצדורה מוחקת את כלל מה שיש על המסך  
מדפיסה את המבוך הראשון וצובעת אותו בורוד- את השלב הראשון

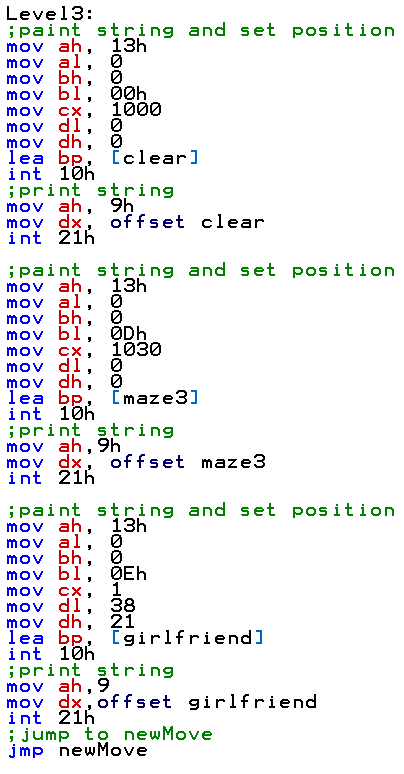
וגם מדפיסה את girlfriend



**פרוצדורת Level2:**

הפרוצדורה מוחקת את כלל מה שיש על המסך  
מדפיסה את המבוך השני וצובעת אותו בוורוד- את השלב השני

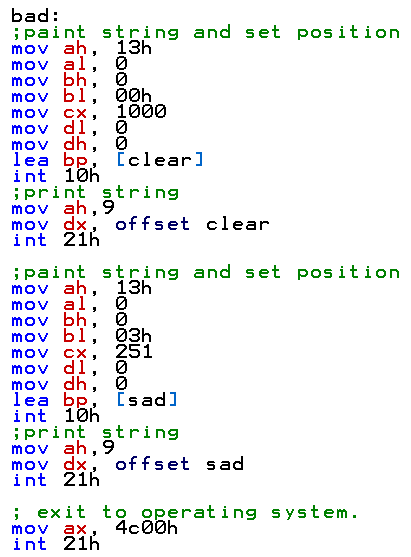
וגם מדפיסה את girlfriend



**פרוצדורת Level3:**

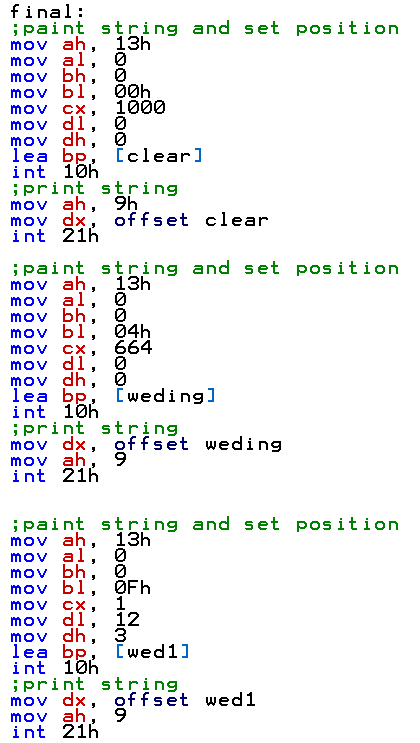
הפרוצדורה מוחקת את כלל מה שיש על המסך  
מדפיסה את המבוך השלישי וצובעת אותו בוורוד- את השלב השלישי

וגם מדפיסה את girlfriend



**פרוצדורת bad:**

הפרוצדורה מוחקת את כלל מה שיש על המסך  
ומדפיסה את SAD בצבע תכלת ומסיימת את המשחק



**פרוצדורת final:**

הפרוצדורה מוחקת את כל מה שיש על המסך

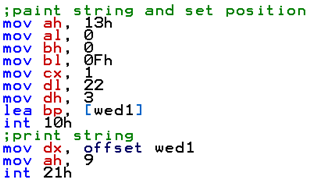
הפרוצדורה מדפיסה את החתונה ואת GIRLFRIEND

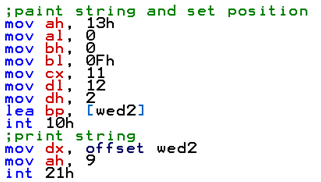
היא צובעת את השביל שנקרא WEDING ואת המסגרת באדום

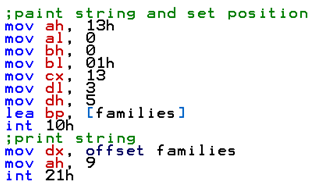
ואת החופה בלבן שנקראת WED1 וWED2

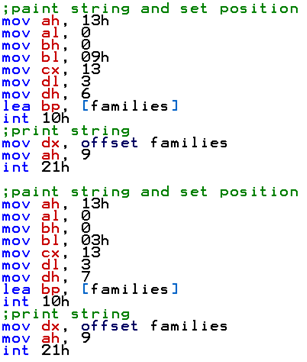
ואחרי זה מתפיסה קבוצות של סמיילים בצבעים שונים במקומות שונים

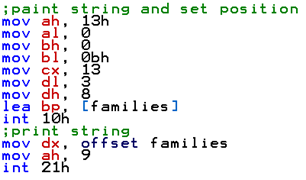
ולאחר מכן את השחקן

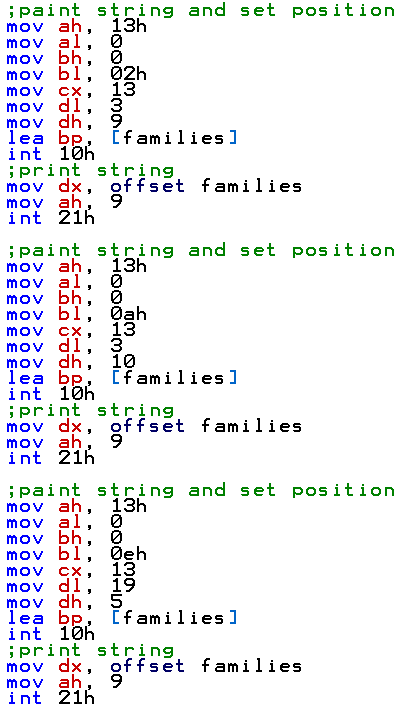


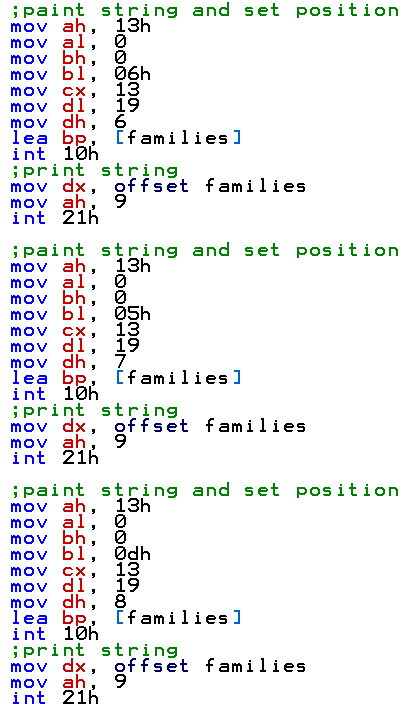


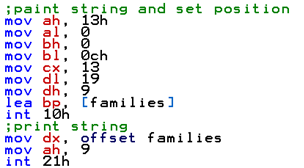


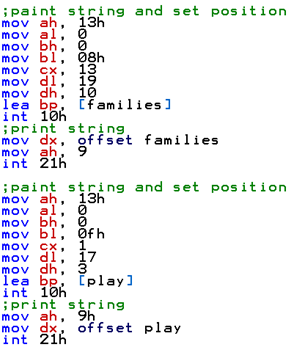




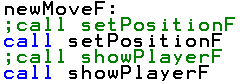


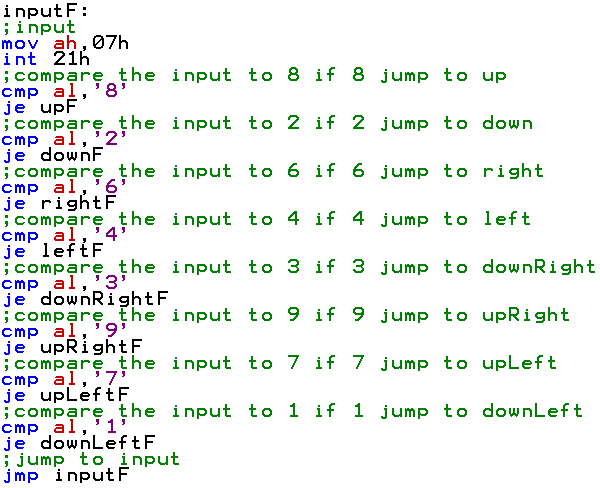




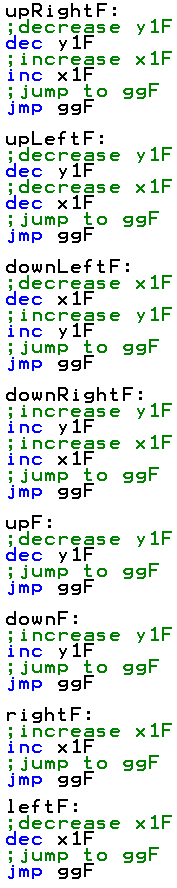


**פרוצדורת newMoveF:**פרוצדורה זו קוראת לפרוצדורות:  
setPositionF  
shopPlayerF





**פרוצדורת inputF:**פרוצדורה זו קולטת תווים וקופצת בהתאם לערכם



**פרוצדורת:**uprightF- מזיזה את הכלה באלכסון ימין למעלה  
upLeftF- מזיזה את הכלה באלכסון שמאל למעלה

downLeftF- מזיזה את הכלה באלכסון שמאל למטה

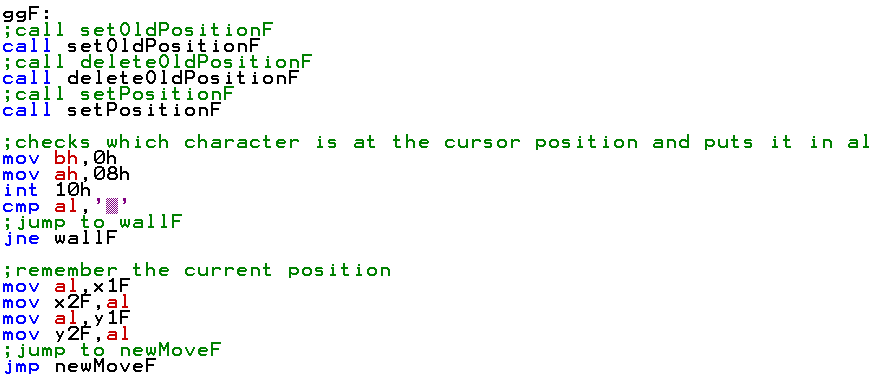
downRightF- מזיזה את הכלה באלכסון ימין למטה

UpF- מזיזה את הכלה למעלה

downF מזיזה את הכלה למטה

RightF- מזיזה את הכלה ימינה

LeftF- מזיזה את הכלה שמאלה



**פרוצדורת ggF:**

קורא ל:

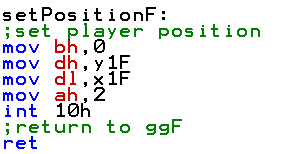
setOldPositionF

deleteOldPositionF

setPositionF

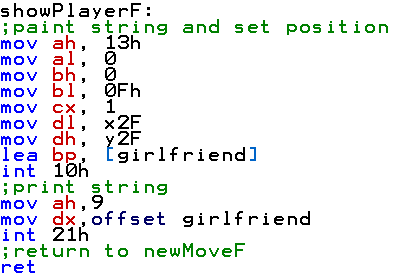
ובודק איפה נמצאת הכלה, אם הכלה במקום ריק הוא קופץ לwallF

והפרוצדורה מכניסה את הערך שבX1F לX2F ושל Y1F לY2F



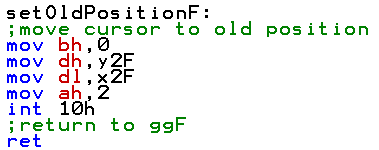
**פרוצדורת setPositionF:**

הפרוצדורה מגדירה את המיקום של הכלה



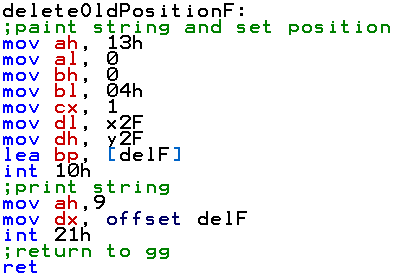
**פרוצדורת showPlayerF:**

הפרוצדורה מדפיסה את הכלה במיקום של X1F Y1F וצובעת אותה בלבן



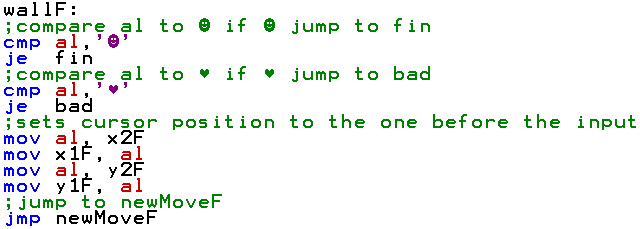
**פרוצדורת setlOldPositionF:**

הפרוצדורה מגדירה את המיקום הישן של הכלה



**פרוצדורת deleteOldPositionF:**

מדפיסה את מה שהיה במיקום הקודם לפני שהכלה הייתה עליו

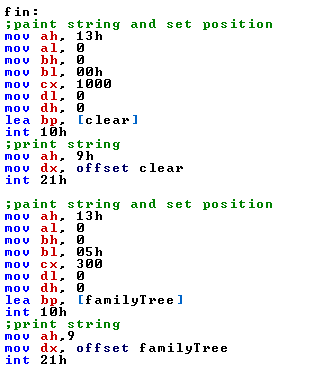


**פרוצדורת wallF:**

בודק איפה נמצאת הכלה  
אם הכלה נמצאת על סמיילי, יקפוץ לlevelsF

אם הכלה נמצאת על לב, יקפוץ ל badF

הפרוצדורה שם את הערך של X2F בX1F ושל Y2F בY1F



**פרוצדורת inputW:**

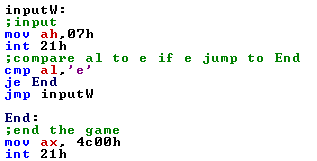
הפרוצדורה קולטת תו ומשווה אותו לe אם הוא e היא קופצת לEnd

**פרוצדורת fin:**

מוחקת את כל מה שיש על המסך

ומדפיסה את familyTree באדום

ויוצאת מהמשחק



**פרוצדורת End:**

הפרוצדורה יוצאת מהמשחק

**תרשים זרימה:**

הדפסת התחלת המשחק

קליטת תו

לא

הדפסת עלילת המשחק

Al=i

Al=g

לא כן

כן

קליטת תו

הדפסת שלב הראשון

Al=c

לא

כן

הדפסת השחקן   
בשורה 1 ובעמודה 1

הדפת הוראות המשחק

קליטת תו

לא

Al=g

כן

כן לא

האם השחקן נמצא על לב

קליטת תו

האם השחקן הגיע ליעד

Al=8

ן כן

הזז אתת השחקן למעלה

לא

Al=2

כן

הזז אתת השחקן למעטה

כן

לא

הזז אתת השחקן ימינה

Al=6

כן

הדפסת שלב השני

לא

הזז אתת השחקן שמאלה

Al=4

כן

הדפסת השחקן   
בשורה 1 ובעמודה 1

לא

הזז את השחקן באלכסון למטה ימינה

Al=3

כן

לא

Al=9

כן

הזז את השחקן באלכסון למעלה ימינה

לא

Al=7

הזז אתת השחקן באלכסון למעלה שמאלה

כן

כן לא

הזז אתת השחקן באלכסון למטה שמאלה

Al=1

כן לא

האם השחקן נמצא על לב

קליטת תו

האם השחקן הגיע ליעד

Al=8

ן כן

הזז אתת השחקן למעלה

לא

Al=2

כן

הזז אתת השחקן למעטה

כן

לא

הזז אתת השחקן ימינה

Al=6

כן

הדפסת שלב השלישי

לא

הזז אתת השחקן שמאלה

Al=4

כן

הדפסת השחקן   
בשורה 1 ובעמודה 1

לא

הזז את השחקן באלכסון למטה ימינה

Al=3

כן

לא

Al=9

כן

הזז את השחקן באלכסון למעלה ימינה

לא

Al=7

הזז אתת השחקן באלכסון למעלה שמאלה

כן

כן לא

הזז אתת השחקן באלכסון למטה שמאלה

Al=1

כן לא

האם השחקן נמצא על לב

קליטת תו

האם השחקן הגיע ליעד

Al=8

ן כן

הזז אתת השחקן למעלה

לא

Al=2

כן

הזז אתת השחקן למעטה

כן

לא

הזז אתת השחקן ימינה

Al=6

כן

הדפסת שלב החתונה

לא

הזז אתת השחקן שמאלה

Al=4

כן

הדפסת השחקן   
בשורה 2 ובעמודה 15

לא

הזז את השחקן באלכסון למטה ימינה

Al=3

כן

לא

Al=9

כן

הזז את השחקן באלכסון למעלה ימינה

לא

Al=7

הזז אתת השחקן באלכסון למעלה שמאלה

כן

כן לא

הזז אתת השחקן באלכסון למטה שמאלה

Al=1

כן לא

האם השחקן נמצא על לב

קליטת תו

האם השחקן הגיע ליעד

Al=8

ן כן

הזז אתת השחקן למעלה

לא

Al=2

כן

הזז אתת השחקן למעטה

כן

לא

הזז אתת השחקן ימינה

Al=6

כן

הדפסת סוף המשחק

לא

הזז אתת השחקן שמאלה

Al=4

כן

קליטת תו

לא

הזז את השחקן באלכסון למטה ימינה

Al=3

לא כן

Al=E

לא

Al=9

כן

הזז את השחקן באלכסון למעלה ימינה

כן

לא

Al=7

הזז אתת השחקן באלכסון למעלה שמאלה

כן

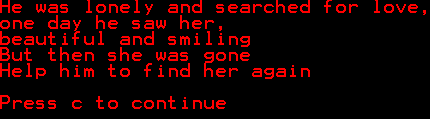
כן לא

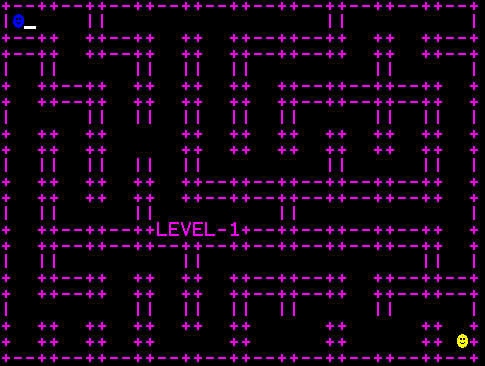
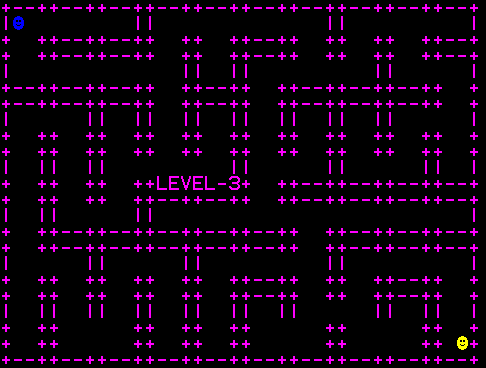
הזז אתת השחקן באלכסון למטה שמאלה

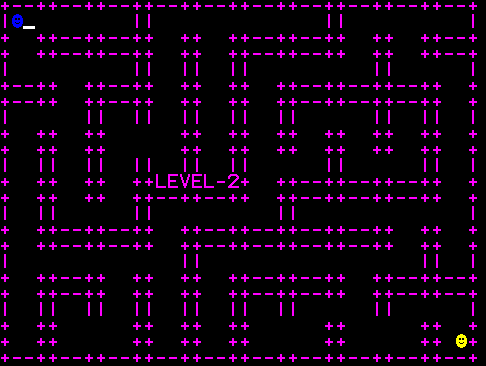
Al=1

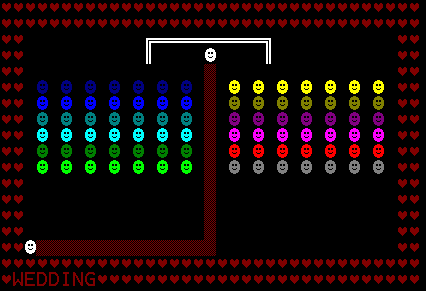
הדפסת את סיום המשחק בהפסד

**תמונות מסך של המשחק:**



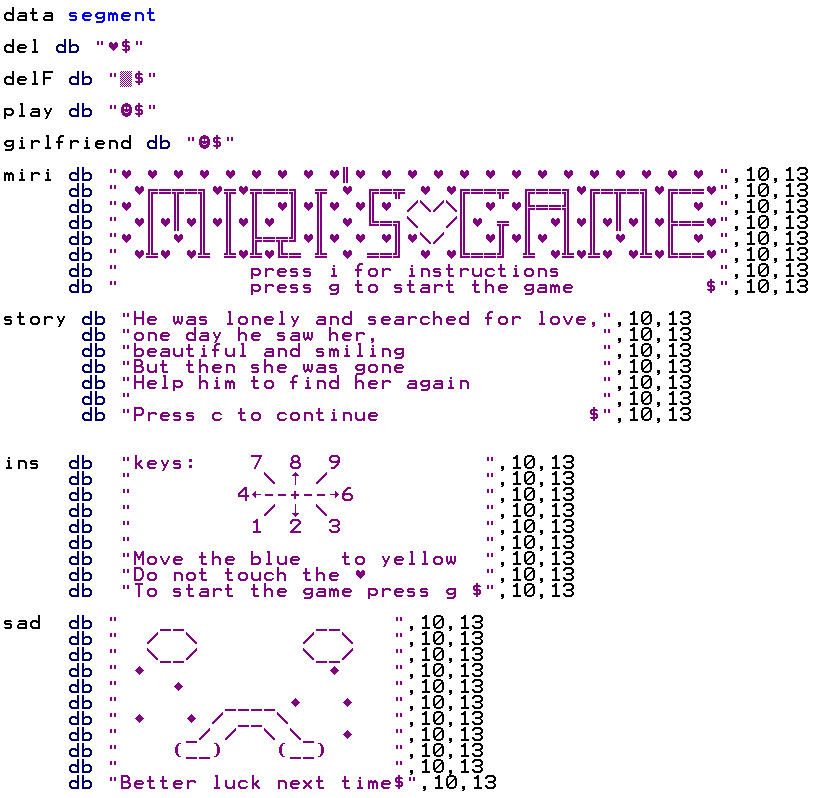


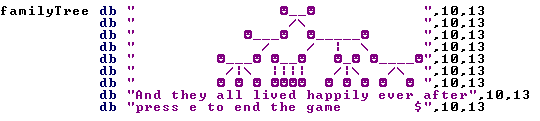


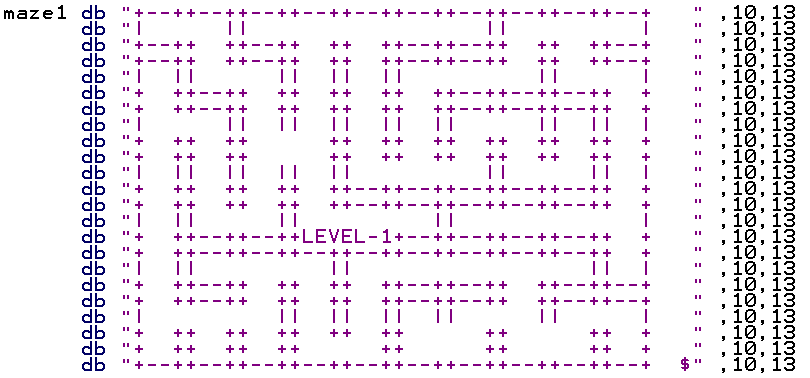


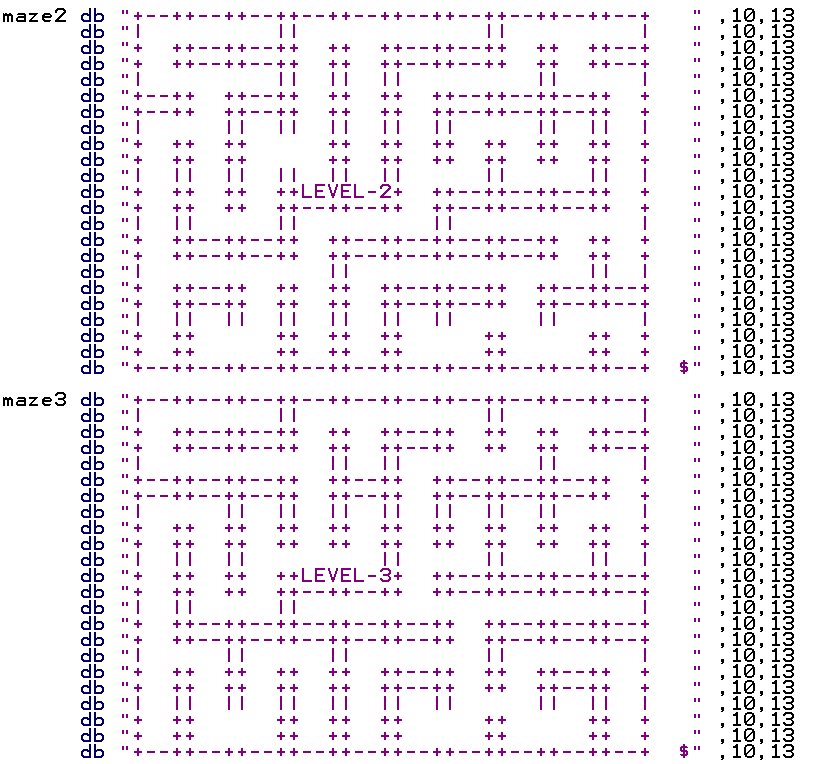


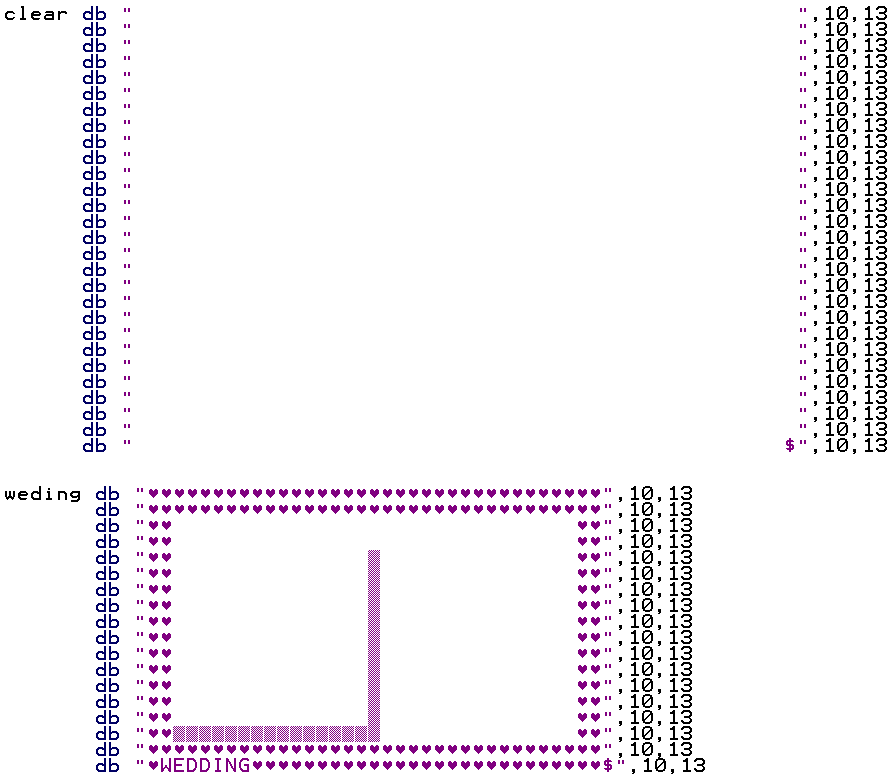
**קוד המשחק:**













y1 db 1

x1 db 1

y2 db 1

x2 db 1

level db 0

y1F db 15

x1F db 2

y2F db 15

x2F db 2

ends

stack segment

dw 128 dup(0)

ends

code segment

start:

; set segment registers:

mov ax, data

mov ds, ax

mov es, ax

GAMESTART:

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0ch

mov cx, 289

mov dl, 0

mov dh, 0

lea bp, [miri]

int 10h

;write string

mov ah, 9h

mov dx, offset miri

int 21h

call WAITING

WAITING:

;input

mov ah,07h

int 21h

;compare input to i if i jump to StoryOfTheGame

cmp al,'i'

je StoryOfTheGame

;compare input to g if g jump to levels

cmp al,'g'

je levels

;jump to WAITING

jmp WAITING

StoryOfTheGame:

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 00h

mov cx, 1000

mov dl, 0

mov dh, 0

lea bp, [clear]

int 10h

;write string

mov ah, 9h

mov dx, offset clear

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0ch

mov cx, 250

mov dl, 0

mov dh, 0

lea bp, [story]

int 10h

;write string

mov ah, 9h

mov dx, offset story

int 21h

storyI:

;input

mov ah,07h

int 21h

;compare input to c if c jump to inst

cmp al,'c'

je inst

;jump to storyI

jmp storyI

inst:

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 00h

mov cx, 1000

mov dl, 0

mov dh, 0

lea bp, [clear]

int 10h

;write string

mov ah, 9h

mov dx, offset clear

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0ch

mov cx, 257

mov dl, 0

mov dh, 0

lea bp, [ins]

int 10h

;write string

mov ah, 9h

mov dx, offset ins

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 01h

mov cx, 1

mov dl, 14

mov dh, 6

lea bp, [play]

int 10h

;write string

mov ah, 9h

mov dx, offset play

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0Eh

mov cx, 1

mov dl, 26

mov dh, 6

lea bp, [girlfriend]

int 10h

;write string

mov ah, 9h

mov dx, offset girlfriend

int 21h

;call inputI

call inputI

inputI:

;input

mov ah,07h

int 21h

;jump to Levels if the input is 'g'

cmp al,'g'

je Levels

;jump to inputI if there is no input

jmp inputI

levels:

;move to x1 x2 y1 y2: 1

mov x1,1

mov x2,1

mov y1,1

mov y2,1

;increase level

inc level

;compare level to 1 if 1 jump to level 1

cmp level,1

je level1

;compare level to 2 if 1 jump to level 2

cmp level,2

je level2

;compare level to 3 if 1 jump to level 3

cmp level,3

je level3

;compare level to 4 if 1 jump to final

cmp level,4

je final

newMove:

;go to setPosition

call setPosition

;go to showPlayer

call showPlayer

input:

;input

mov ah,07h

int 21h

;compare the input to 8 if 8 jump to up

cmp al,'8'

je up

;compare the input to 2 if 2 jump to down

cmp al,'2'

je down

;compare the input to 6 if 6 jump to right

cmp al,'6'

je right

;compare the input to 4 if 4 jump to left

cmp al,'4'

je left

;compare the input to 3 if 3 jump to downRight

cmp al,'3'

je downRight

;compare the input to 9 if 9 jump to upRight

cmp al,'9'

je upRight

;compare the input to 7 if 7 jump to upLeft

cmp al,'7'

je upLeft

;compare the input to 1 if 1 jump to downLeft

cmp al,'1'

je downLeft

;jump to input

jmp input

upRight:

;decrease y1

dec y1

;increase x1

inc x1

;jump to gg

jmp gg

upLeft:

;decrease y1

dec y1

;decrease x1

dec x1

;jump to gg

jmp gg

downLeft:

;decrease x1

dec x1

;increase y1

inc y1

;jump to gg

jmp gg

downRight:

;increase y1

inc y1

;increase x1

inc x1

;jump to gg

jmp gg

up:

;decrease y1

dec y1

;jump to gg

jmp gg

down:

;increase y1

inc y1

;jump to gg

jmp gg

right:

;increase x1

inc x1

;jump to gg

jmp gg

left:

;decrease x1

dec x1

;jump to gg

jmp gg

gg:

;call to setOldPosition

call setOldPosition

;call to deleteOldPosition

call deleteOldPosition

;call to setPosition

call setPosition

;checks which character is at the cursor position and puts it in al

mov bh,0h

mov ah,08h

int 10h

cmp al,' '

jne wall

;remember the current position

mov al,x1

mov x2,al

mov al,y1

mov y2,al

;jump to newMove

jmp newMove

setPosition:

;set player position

mov bh,0

mov dh,y1

mov dl,x1

mov ah,2

int 10h

;return to gg

ret

showPlayer:

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 09h

mov cx, 1

mov dl, x2

mov dh, y2

lea bp, [play]

int 10h

;print string

mov ah,9

mov dx,offset play

int 21h

;return to newMove

ret

setOldPosition:

;move cursor to old position

mov bh,0

mov dh,y2

mov dl,x2

mov ah,2

int 10h

;return to gg

ret

deleteOldPosition:

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 4h

mov cx, 1

mov dl, x2

mov dh, y2

lea bp, [del]

int 10h

;print string

mov ah,9

mov dx, offset del

int 21h

;return to gg

ret

wall:

;compare al to if jump to levels

cmp al,''

je levels

;compare al to if jump to bad

cmp al,''

je bad

;sets cursor position to the one before the input

mov al, x2

mov x1, al

mov al, y2

mov y1, al

;jump to newMove

jmp newMove

Level1:

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 00h

mov cx, 1000

mov dl, 0

mov dh, 0

lea bp, [clear]

int 10h

;print string

mov ah, 9h

mov dx, offset clear

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0Dh

mov cx, 1030

mov dl, 0

mov dh, 0

lea bp, [maze1]

int 10h

;print string

mov ah,9h

mov dx, offset maze1

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0Eh

mov cx, 1

mov dl, 38

mov dh, 21

lea bp, [girlfriend]

int 10h

;print string

mov ah,9

mov dx,offset girlfriend

int 21h

;jump to newMove

jmp newMove

Level2:

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 00h

mov cx, 1000

mov dl, 0

mov dh, 0

lea bp, [clear]

int 10h

;print string

mov ah, 9h

mov dx, offset clear

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0Dh

mov cx, 1030

mov dl, 0

mov dh, 0

lea bp, [maze2]

int 10h

;print string

mov ah,9h

mov dx, offset maze2

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0Eh

mov cx, 1

mov dl, 38

mov dh, 21

lea bp, [girlfriend]

int 10h

;print string

mov ah,9

mov dx,offset girlfriend

int 21h

;jump to newMove

jmp newMove

Level3:

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 00h

mov cx, 1000

mov dl, 0

mov dh, 0

lea bp, [clear]

int 10h

;print string

mov ah, 9h

mov dx, offset clear

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0Dh

mov cx, 1030

mov dl, 0

mov dh, 0

lea bp, [maze3]

int 10h

;print string

mov ah,9h

mov dx, offset maze3

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0Eh

mov cx, 1

mov dl, 38

mov dh, 21

lea bp, [girlfriend]

int 10h

;print string

mov ah,9

mov dx,offset girlfriend

int 21h

;jump to newMove

jmp newMove

bad:

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 00h

mov cx, 1000

mov dl, 0

mov dh, 0

lea bp, [clear]

int 10h

;print string

mov ah,9

mov dx, offset clear

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 03h

mov cx, 251

mov dl, 0

mov dh, 0

lea bp, [sad]

int 10h

;print string

mov ah,9

mov dx, offset sad

int 21h

; exit to operating system.

mov ax, 4c00h

int 21h

final:

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 00h

mov cx, 1000

mov dl, 0

mov dh, 0

lea bp, [clear]

int 10h

;print string

mov ah, 9h

mov dx, offset clear

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 04h

mov cx, 664

mov dl, 0

mov dh, 0

lea bp, [weding]

int 10h

;print string

mov dx, offset weding

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0Fh

mov cx, 1

mov dl, 12

mov dh, 3

lea bp, [wed1]

int 10h

;print string

mov dx, offset wed1

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0Fh

mov cx, 1

mov dl, 22

mov dh, 3

lea bp, [wed1]

int 10h

;print string

mov dx, offset wed1

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0Fh

mov cx, 11

mov dl, 12

mov dh, 2

lea bp, [wed2]

int 10h

;print string

mov dx, offset wed2

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 01h

mov cx, 13

mov dl, 3

mov dh, 5

lea bp, [families]

int 10h

;print string

mov dx, offset families

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 09h

mov cx, 13

mov dl, 3

mov dh, 6

lea bp, [families]

int 10h

;print string

mov dx, offset families

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 03h

mov cx, 13

mov dl, 3

mov dh, 7

lea bp, [families]

int 10h

;print string

mov dx, offset families

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0bh

mov cx, 13

mov dl, 3

mov dh, 8

lea bp, [families]

int 10h

;print string

mov dx, offset families

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 02h

mov cx, 13

mov dl, 3

mov dh, 9

lea bp, [families]

int 10h

;print string

mov dx, offset families

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0ah

mov cx, 13

mov dl, 3

mov dh, 10

lea bp, [families]

int 10h

;print string

mov dx, offset families

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0eh

mov cx, 13

mov dl, 19

mov dh, 5

lea bp, [families]

int 10h

;print string

mov dx, offset families

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 06h

mov cx, 13

mov dl, 19

mov dh, 6

lea bp, [families]

int 10h

;print string

mov dx, offset families

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 05h

mov cx, 13

mov dl, 19

mov dh, 7

lea bp, [families]

int 10h

;print string

mov dx, offset families

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0dh

mov cx, 13

mov dl, 19

mov dh, 8

lea bp, [families]

int 10h

;print string

mov dx, offset families

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0ch

mov cx, 13

mov dl, 19

mov dh, 9

lea bp, [families]

int 10h

;print string

mov dx, offset families

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 08h

mov cx, 13

mov dl, 19

mov dh, 10

lea bp, [families]

int 10h

;print string

mov dx, offset families

mov ah, 9

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0fh

mov cx, 1

mov dl, 17

mov dh, 3

lea bp, [play]

int 10h

;print string

mov ah, 9h

mov dx, offset play

int 21h

newMoveF:

;call setPositionF

call setPositionF

;call showPlayerF

call showPlayerF

inputF:

;input

mov ah,07h

int 21h

;compare the input to 8 if 8 jump to upF

cmp al,'8'

je upF

;compare the input to 2 if 2 jump to downF

cmp al,'2'

je downF

;compare the input to 6 if 6 jump to rightF

cmp al,'6'

je rightF

;compare the input to 4 if 4 jump to leftF

cmp al,'4'

je leftF

;compare the input to 3 if 3 jump to downRightF

cmp al,'3'

je downRightF

;compare the input to 9 if 9 jump to upRightF

cmp al,'9'

je upRightF

;compare the input to 7 if 7 jump to upLeftF

cmp al,'7'

je upLeftF

;compare the input to 1 if 1 jump to downLeftF

cmp al,'1'

je downLeftF

;jump to input

jmp inputF

upRightF:

;decrease y1F

dec y1F

;increase x1F

inc x1F

;jump to ggF

jmp ggF

upLeftF:

;decrease y1F

dec y1F

;decrease x1F

dec x1F

;jump to ggF

jmp ggF

downLeftF:

;decrease x1F

dec x1F

;increase y1F

inc y1F

;jump to ggF

jmp ggF

downRightF:

;increase y1F

inc y1F

;increase x1F

inc x1F

;jump to ggF

jmp ggF

upF:

;decrease y1F

dec y1F

;jump to ggF

jmp ggF

downF:

;increase y1F

inc y1F

;jump to ggF

jmp ggF

rightF:

;increase x1F

inc x1F

;jump to ggF

jmp ggF

leftF:

;decrease x1F

dec x1F

;jump to ggF

jmp ggF

ggF:

;call setOldPositionF

call setOldPositionF

;call deleteOldPositionF

call deleteOldPositionF

;call setPositionF

call setPositionF

;checks which character is at the cursor position and puts it in al

mov bh,0h

mov ah,08h

int 10h

cmp al,'±'

;jump to wallF

jne wallF

;remember the current position

mov al,x1F

mov x2F,al

mov al,y1F

mov y2F,al

;jump to newMoveF

jmp newMoveF

setPositionF:

;set player position

mov bh,0

mov dh,y1F

mov dl,x1F

mov ah,2

int 10h

;return to ggF

ret

showPlayerF:

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 0Fh

mov cx, 1

mov dl, x2F

mov dh, y2F

lea bp, [girlfriend]

int 10h

;print string

mov ah,9

mov dx,offset girlfriend

int 21h

;return to newMoveF

ret

setOldPositionF:

;move cursor to old position

mov bh,0

mov dh,y2F

mov dl,x2F

mov ah,2

int 10h

;return to ggF

ret

deleteOldPositionF:

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 04h

mov cx, 1

mov dl, x2F

mov dh, y2F

lea bp, [delF]

int 10h

;print string

mov ah,9

mov dx, offset delF

int 21h

;return to ggF

ret

wallF:

;compare al to if jump to fin

cmp al,''

je fin

;compare al to if jump to bad

cmp al,''

je bad

;sets cursor position to the one before the input

mov al, x2F

mov x1F, al

mov al, y2F

mov y1F, al

;jump to newMoveF

jmp newMoveF

fin:

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 00h

mov cx, 1000

mov dl, 0

mov dh, 0

lea bp, [clear]

int 10h

;print string

mov ah, 9h

mov dx, offset clear

int 21h

;paint string and set position

mov ah, 13h

mov al, 0

mov bh, 0

mov bl, 05h

mov cx, 300

mov dl, 0

mov dh, 0

lea bp, [familyTree]

int 10h

;print string

mov ah,9

mov dx, offset familyTree

int 21h

inputW:

;input

mov ah,07h

int 21h

;compare al to e if e jump to End

cmp al,'e'

je End

jmp inputW

End:

;end the game

mov ax, 4c00h

int 21h

ends

end start ; set entry point and stop the assembler.

**סיכום ותודות:**

מאוד נהניתי בבניית משחק זה וגיליתי שאני מסוגלת לבנות משחק בכוחות עצמי (עם קצת עזרה) ולהגיע לתוצאה שרציתי.

באמצעות משחק זה גם הוכחתי לעצמי, לחבריי ולמשפחתי שגם אישה (אני הבת היחידה במגמת מחשבים, מתוך 5 בנות אני היחידה שלא נשרה מלימודי מדעי המחשב) יכולה להצליח במגמת מחשבים ואפילו להגיע לפרויקט זה שזו הפרויקט האחרון שמשלים לי ל5 יחידות מדעי המחשב (כמובן אני אעשה עוד 5 יחידות שישלים לי ל10 יחידות מדעי המחשב) .

אני מודעת שאפשר היה לקצר את המשחק ולעלות את יעילותו אך בשל חוסר זמן העליתי רק קצת מיעילותו.

בהתחלה היה לי מאוד קשה בבניית המשחק כי לא ידעתי את הבסיס של איך להזיז דברים ולצבוע אך באמצעות עזרת חברי, המורה שלומי, ואתרים באינטרנט עם הסברים הצלחתי לבנות את המשחק הזה ותוך שבוע הכפלתי את הצלחת המשחק שלי פי שתיים (ממשחק שלא ידעתי איך להזיז בו את השחקנים למשחק צבעוני עם שחקנים זזים).

**ביוגרפיה:**

<http://www.coval.net/teach/emu8086/helpfiles/supported_interrupts.html>

האתר שלקחתי ממנו את הפסיקות

<http://stackoverflow.com/questions/1575530/how-could-i-stop-from-printing-both-sides-of-a-wall-in-my-ascii-maze>

האתר שלקחתי ממנו רעיונות למבוכים

<http://patorjk.com/software/taag/#p=display&f=Graffiti&t=Type%20Something%20>

האתר שממנו לקחתי רעיונות לעיצובים