**פרויקט גמר אסמבלי**

**HangMan**

**שם המשחק:** HangMan

**מגיש:** ליאור שם טוב

**כיתה:** יא'6

**מורה:** איציק בן שטרית

**שנה:** 2024

**בית ספר:** מקיף ז'



תוכן עניינים:

עמוד 2 המשחק HangMan

עמודים 4 - 8 סביבת העבודה

עמוד 9 הגדרת משתנים

עמוד 10 תת משימות של הפרויקט

עמודים 11 - 12 תתי תוכניות

עמודים 13 - 28 אלגוריתם

עמודים 29 - 58 יישום

עמודים 59 - 64 תמונות מהתוכנית

# 

**המשחק HangMan:**

**מהות המשחק**:

במשחק HangMan, כולל שחקן אחד שמטרתו לנחש מילה נסתרת לפני שמסיימים "לתלות" את האיש.

**תחילת המשחק**:

השחקן מקבל נושא שמרמז על המילה (לדוגמה: "חיות", "מדינות").

המילה מופיעה כמספר קווים תחתונים, שכל אחד מהם מייצג אות חסרה במילה.

**מהלך המשחק**:

בכל תור השחקן מנחש אות אחת (a–z).

אם האות נכונה, היא נחשפת במיקומה במילה.

אם האות שגויה, האיש התלוי מתקרב שלב נוסף לסיומו (נוסף חלק לגופו), ומספר הנסיונות יורד.

**המטרה:**

לנחש את המילה במלואה לפני שכל חלקי הגוף של האיש התלוי מוצגים.

**סיום:**

אם המילה נחשפה לפני שהאיש "נתלה", השחקן ניצח.

אם לא, מוצגת המילה והמשחק מסתיים בהפסד.



**סביבת העבודה:**

שפת התוכנה: Assembly

סביבת העבודה: emu8086

סביבת העבודה **Emu8086** פותחה על ידי חברת **EMULATOR8086 LLC**. החברה מתמקדת ביצירת כלים ללימוד ותרגול שפת Assembly, עם דגש על מעבד 8086.

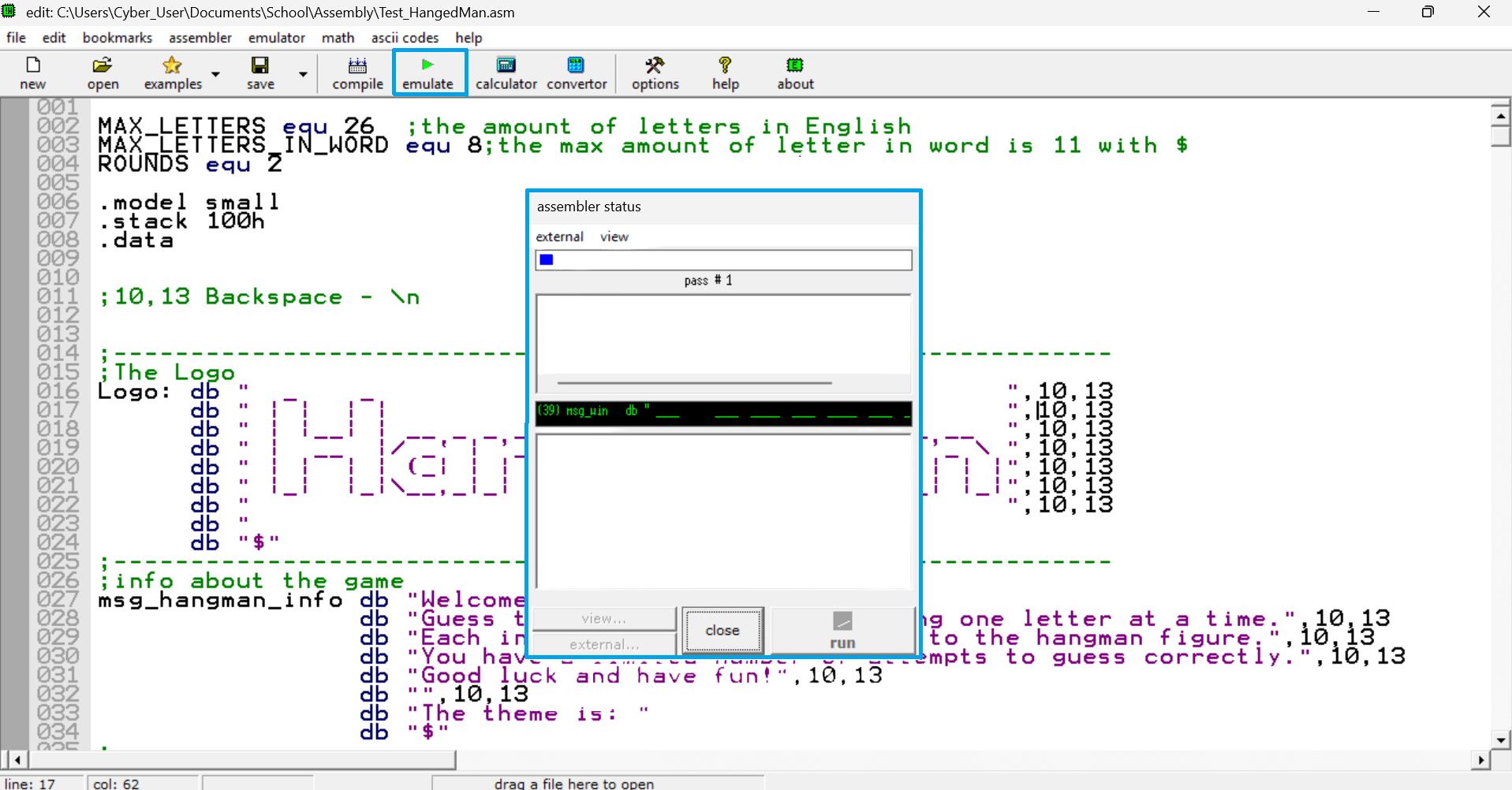
בסביבת העבודה ניתן לכתוב קוד בשפת אסמבלי.

ניתן לקמפל ולהריץ אותו על המחשב.

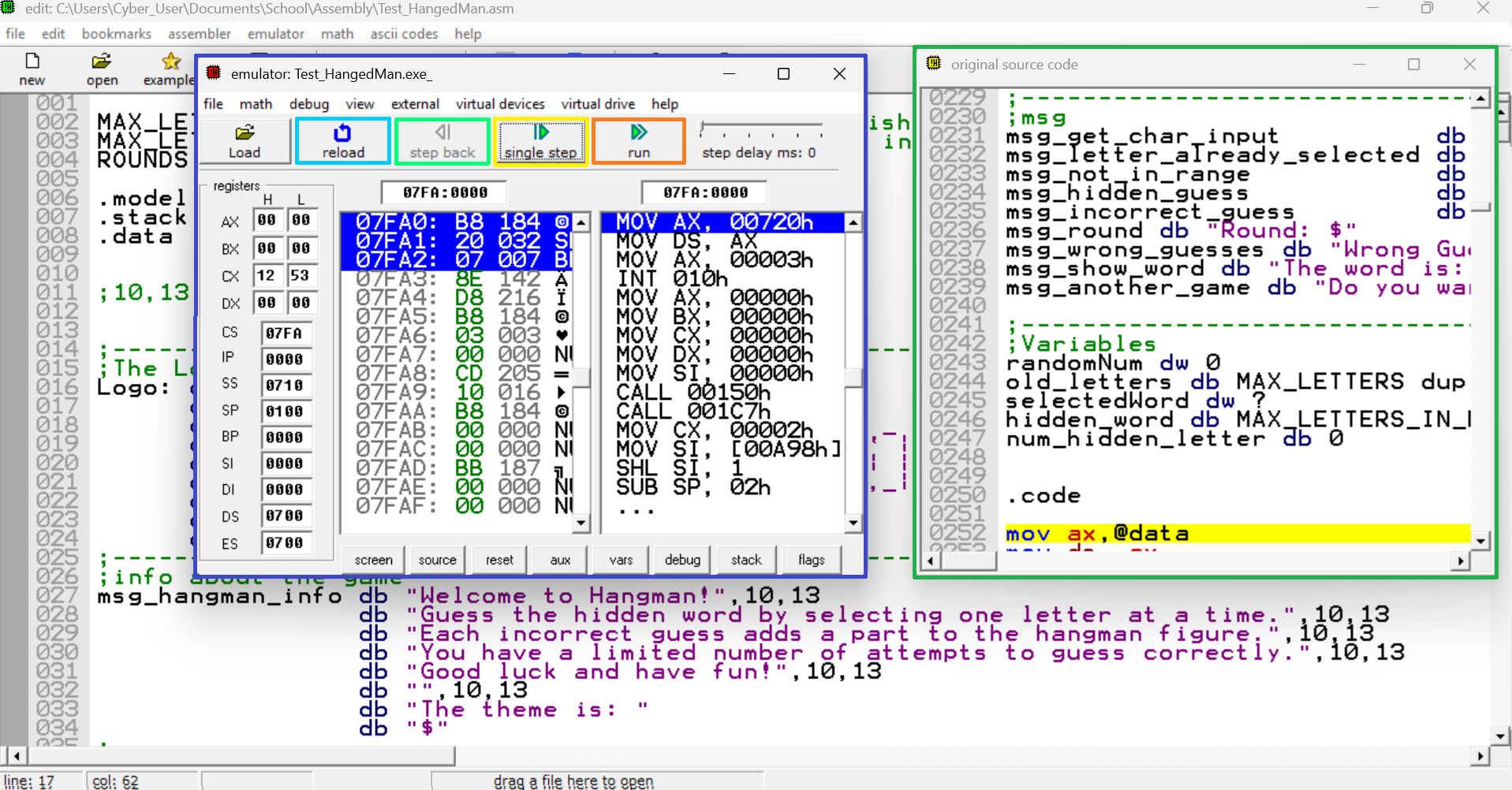
בסביבת העבודה ניתן לראות את מצב ה - CPU לאחר כל פקודה שביצענו או לאחר מספר פקודות.

צילומים של סביבת העבודה:

בשביל להריץ את הקוד צריך ללחוץ על **emulate**כשלוחצים על emulate נפתח לנו המקמפל של הקוד



אם הקוד עובר את שלב הקומפילציה בהצלחה נפתחות שני החלונות האלה:

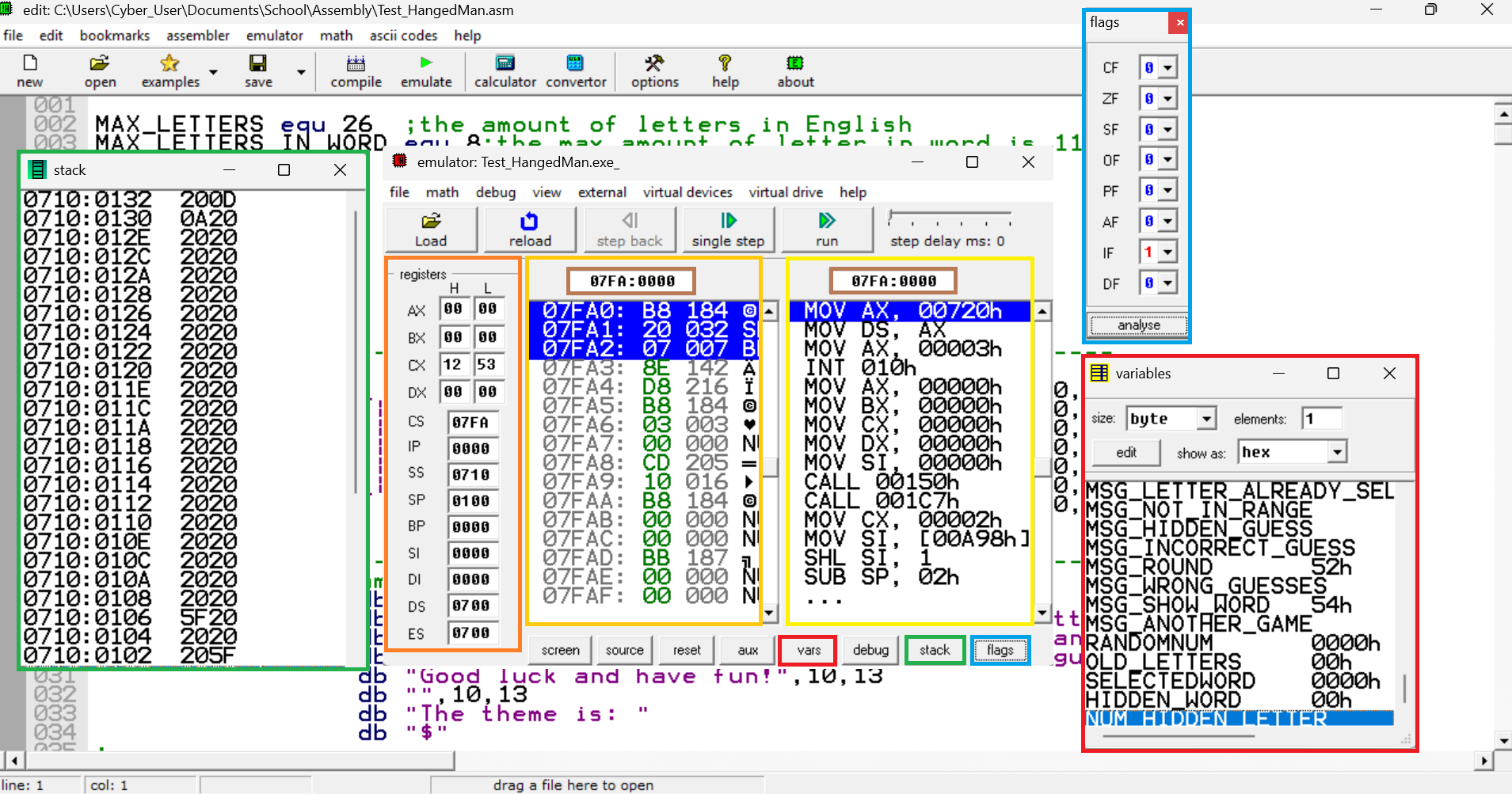


חלון כחול:

* להרית את כל הקוד
* להריץ שורה אחת
* להחזיר שורה אחורה
* להריץ את הקוד מחדש

חלון ירוק:

* מופיע שורות הקוד של התוכנית
* הפס הצהוב הוא ה ip שמצביע על הפעולה הבאה לביצוע
* רשימת האוגרים
* הקוד בשפת מכונה
* הקוד באסמבלי
* רשימת הדגלים
* המחסנית
* רשימת משתנים
* מיקום הקוד בזכרון

****

מקטעים:

גודל כל מקטע הוא 64k.

Code segment - מקטע הקוד - יכיל את הקוד של התוכנה.

Data segment - מקטע הנתונים - ישמור את כל הנתונים של כל התוכנית.

Stack segment - מקטע המחסנית.

Extended segment - מקטע הרזרבי (אם אין מספיק מקום מקום במקטעים האחרים משתמשים בו).

הדגלים:

FR - אוגר הדגלים, בן 16 סיביות המכיל 10 דגלים.

השתמשתי ב 3 דגלים:

S - דגל הסימן - נדלק שתוצאת פעולה אריתמטית היא שלילית.

Z - דגל האפס - נדלק שתוצאת פעולה אריתמטית היא 0.

C - נדלק שיש שארית או שפעולה אריתמטית חורגת מתחום הזיכרון.

האוגרים:  
האוגרים יכולים להיות אחדות ואפסים ארבעת האוגר המרכזיים הם 16 סיביות. שמתפרקים לשני אוגרים:

אוגר L (נמוך) 0-7 סיביות בזיכרון.

אוגר H (גבוה) 8-15 סיביות בזיכרון.

AX = AH,AL

BX = BH,HL

CX = CH,CL

DX = DH,DL

IP - אוגר בן 16 סיביות - מצביע על הכתובת של ההוראה הבאה לביצוע.

IR - אוגר בן 8 סיביות - מכיל את הבית הראשון של ההוראה שמתבצעת (הוראה יכולה להכיל יותר מבית 1).

המחסנית:

המחסנית (Stack) היא מקטע המחסנית שפועל במבנה LIFO. דחיפה (PUSH) מוסיפה ערך לראש המחסנית ומקטינה את SP, ושליפה (POP) מוציאה ערך מראש המחסנית ומגדילה את SP .

SS מצביע על ראש המחסנית ו-SP על הפריט העליון. המחסנית משמשת לאחסון נתונים זמניים, כתובות חזרה וניהול קריאות לפונקציות.

BP - משמש לגישה למידע בתוך מקטע המחסנית(בדומה ל BX ב Data Segment)

ומצביע על סוף המחסנית.

Random באסמבלי:

ב emu8086 בכתובת **046C** במקטע (**0x0040** (BIOS Data Segment נשמר שעון המערכת, שמונה "טיקים" מאז חצות.

ניתן לגשת לערך הזה כדי לקבל מידע על הזמן או לייצר מספרים אקראיים.

או בעזרת הפסקה:

mov ah, 2Ch ; בקשה לקבל את הזמן הנוכחי

int 21h ; קריאה לפסיקה

; CH = שעה, CL = דקות, DH = שניות, DL = מאיות השנייה

**הגדרת המשתנים:**

קבועים(equ):

**MAX\_LETTERS** - מציין את מספר האותיות באנגלית

**MAX\_LETTERS\_IN\_WORD** - מציין את כמות האותיות המקסימלית שיש במילה

**ROUNDS** - מציין את כמות הסיבובים

משתנים המכילים הודעות(מחרוזות) להדפסה:

**Logo** - מכיל את הלוגו בתוי אסקי

**msg\_hangman\_info** - מכיל את הסבר המשחק

**msg\_win** - מכיל את הודעת ההפסד בתוי אסקי

**msg\_lose** - מכיל את הודעת ההפסד בתוי אסקי

**msg\_Good\_Bye** - מכיל גוד ביי בתוי אסקי

**msg\_get\_char\_input**  - מכיל הכנס תו

**msg\_letter\_already\_selected**  - מכיל האות כבר נבחרה

**msg\_not\_in\_range** - מכיל אות לא טווח

**msg\_hidden\_guess** - מכיל ניחשת אות נכונה

**msg\_incorrect\_guess** - מכיל ניחוש לא נכון בהצלחה בפעם הבאה

**msg\_round** - מכיל ראונד

**msg\_wrong\_guesses** - מכיל ניחושים לא נכונים

**msg\_show\_word** - מכיל המילה היא

**msg\_another\_game** - אתה רוצה עוד משחק

משתנים שהם לא להדפסה:

**fruits0 - fruits19**  - מכילים מילים מהנושא פרות

**animals0 - animals19** - מכילים מילים מהנושא חיות

**cities0 - cities19** - מכיל מילים מהנושא ערים

**Hanged\_man0 - Hanged\_man6**  - מכילים את כל האפשרויות של האיש תלוי

**topic0 - topic2** - מכילים את כל הנושאים במשחק

**Hanged\_mans** - מערך של מצביעים של כל האפשרויות של האיש התלוי

**words** - מערך של מצביעים שמכיל את כל המילים האפשריות

**topics** - מערך של מצביעים לכל הנושאים האפשריים

**Hanged\_man\_index** - משתנה 8 סיביות, מכיל כמה פעמים הודפס האיש התלוי

**randomNum** - משתנה 16 סיביות, מכיל את המספר הרנדומלי

**old\_letters** - מערך 8 סיביות ,מכיל את כל האותיות שניחשו

**selectedWord** - משתנה 16 סיביות, מצביע למילה המוגרלת

**hidden\_word** - משתנה 8 סיביות, מכיל את המילה המוגרלת מוסתרת

**num\_hidden\_letter** - משתנה 8 סיביות, מכיל את כמות האותיות הלא מוסתרות

**תת משימות של הפרויקט:**

להדפיס לוגו למסך והסבר על המשחק.

להגריל מילה ונושא רנדומלי.

ולהדפיס נושא רנדומלי.

להדפיס את כמות הסיבובים.

להדפיס את האותיות הלא נכונות.

לעדכן מילה מוסתרת עם "\_" ועם האותיות שניחשו.

להדפיס מילה מוסתרת.

בודק ניצחון(כל האותיות במילה המוסתרת גלויות ).

לקלוט תו מהמשתמש.

לבדוק שהתו הוא אות קטנה באנגלית.

לבדוק שהאות המנוחשת לא נמצא כבר במילים המוסרתרות.

לבדוק שהאות המנוחשת נמצאת במילה .

להדפיס את האיש התלוי בניחוש לא נכון.

להדפיס את המילה.

להדפיס הודעת ניצחון.

להדפיס הודעת הפסד.

להדפיס הודעת סיום(GOOD BYE).

**תתי תוכניות:**

|  |  |  |  |
| --- | --- | --- | --- |
| שם | קלט | פלט | מה מבצעת? |
| menu | כלום | מדפיס לוגו למסך,  מספיס מידע על השמחק, מדפיס נושא מוגרל | מדפיסה לוגו ומידע על המשחק, שם  ב randomNum מספר רנדומלי בין 0-59  מגריל ומדפיס נושא  שם את המילה המוגרלת  ב selectedWord |
| print\_string | מקבל מצביע למחרוזת מהמחסנית | מדפיס את המחרוזת למסך | מדפיס את המילה מהמחסנית למסך, מוסיף 2 למחסנית כדי לנקות את המקום שהתקבל |
| updateHiddenWord | כלום | ממלא את המילה המוסתרת | ממלא את המילה המוסתרת ב "\_" אם עדיין לא ניחשו את האות במילה ובאות שניחשו אותה |
| check\_win | מקום שמור במחסנית | מחזיר 1 אם השחקן ניצח ו 0 אם לא דרך המחסנית | בודק שכל התווים ב hidden\_word הם לא "\_" |
| check\_correct\_guess | מקום שמור במחסנית | מחזיר 1 אם המשתמש ניחש אות נכון אחרת מחזיר 0 | בודק אם כמות התווים שהם לא "\_" גדול מ num\_hidden\_letter |
| get\_input | מקום שמור במחסנית | מחזיר 1 אם האות בין a-z וגם לא נוחשה כבר אחרת מחזיר 0 דרך המחסנית | בודק אם האות היא בין a-z  וגם בודק אם האות לא נוחשה כבר אם היא לא בתנאים מדפיס הודעת שגיאה |
| get\_char | מקום שמור במחסנית | מחזיר את התו שהמשתמש הכניס דרך המחסנית | המשתנש מכניס תו בעזרת פסקה, יורד שורה ומחזיר את התו דרך המחסנית |
| check\_illegal\_letter | מקבל תו דרך המחסנית, שומר מקום במחסנית | מחזיר 1 אם התו בטווח בין a-z אחרת מחזיר 0 דרך המחסנית | בודק אם התו שהוכנס בטווח אסקי בין 97 - 122  אם כן מחזיר 1, אחרת מחזיר 0 דרך המחסנית, מוסיף 2 למחסנית כדי לנקות את המקום שהתקבל |
| check\_letter\_exists | מקבל תו דרך המחסנית, שומר מקום במחסנית | מחזיר 1 אם האות לא נוחשה קבר אחרת מחזיר 0 | בודק אם האות לא נמצאת במערך של האותיות שנחשו  ומחזיר אחד אם האות לא נמצאת, אחרת מחזיר 0 דרך המחסנית, מוסיף 2 למחסנית כדי לנקות את המקום שהתקבל |
| add\_letter | מקבל תו דרך המחסנית | מוסיף אות למערך של הניחושים | מוסיף אות למערך של הניחושים(אחרי האות האחרונה שנוחשה), מוסיף 2 למחסנית כדי לנקות את המקום שהתקבל |
| print\_letter\_guses | כלום | מדפיס את כל האותיות הלא נכונות שניחשו עם רווח בין כל אות | עובר על על המערך של הניחושים ובודק שהאות לא נמצאת במילה שהוגרלה |
| generate\_random\_number | מקבל תו דרך המחסנית, שומר מקום במחסנית | מחזיר את מספר רנדומלי בין 0 - לערך שהוכנס לפונקציה | מקבל את הזמן מהשעון בעזרת פסיקה ומחלק אותו בערך מהמחסנית ומחזיר אותו דרך מהמחסנית, מוסיף 2 למחסנית כדי לנקות את המקום שהתקבל |
| newline | כלום | מדפיס שורה חדשה | מדפיס את התו 10 ואת התו 13 בעזרת פסיקה |

**אלגוריתם:**

קרא לתת תוכנית menu

קרא לתת תוכנית updateHiddenWord

השם ב cx את כמות קבוע ROUNDS

**main\_game:**

שם ב si את Hanged\_man\_index

מכפיל את si ב 2

**courrect\_letter\_loop:**

מחסיר 2 מ sp

מכניס את bp לראש המחסנית

קורא ל check\_win

השם ב bp את sp

השם ב dx את [bp+2] מהחסנית

השם ב bp את הערך בראש המחסנית

מוסיף ל sp את הערך 2

אם dl == 1:

קופץ ל player\_win

קורא ל newline

קורא ל newline

השם ב dx את מצביע ל msg\_round

מכניס את dx לראש המחסנית

קורא ל print\_string

השם ב dl את cl

הוסף ל dl את '0'

השם ב ah את הערך 2

קורא לפונקציה במיקום 21h

קורא ל newline

השם ב dx את מצביע ל msg\_wrong\_guesses

מכניס את dx לראש המחסנית

קורא ל print\_string

קורא ל print\_letter\_guses

קורא ל newline

קורא ל newline

השם ב dx את מצביע ל hidden\_word

מכניס את dx לראש המחסנית

קורא ל print\_string

קורא ל newline

קורא ל newline

**enter\_agin:**

מחסיר 2 מ sp

מכניס את bp לראש המחסנית

קורא ל get\_input

השם ב bp את sp

השם ב dx את [bp+2] מהחסנית

השם ב bp את הערך בראש המחסנית

מוסיף ל sp את הערך 2

אם dl != 1:

קופץ ל enter\_agin

קורא ל updateHiddenWord

קורא ל check\_correct\_guess

מחסיר 2 מ sp

מכניס את bp לראש המחסנית

קורא ל check\_correct\_guess

השם ב bp את sp

השם ב dx את [bp+2] מהחסנית

השם ב bp את הערך בראש המחסנית

מוסיף ל sp את הערך 2

אם dl == 1:

קופץ ל courrect\_letter\_loop

השם ב dx את מצביע ל msg\_incorrect\_guess

מכניס את dx לראש המחסנית

קורא ל print\_string

קורא ל newline

השם ב dx את מצביע ל [Hanged\_mans + si]

מכניס את dx לראש המחסנית

קורא ל print\_string

קורא ל newline

מוסיף אחד ל Hanged\_man\_index

אם cl !=0:

קופץ ל main\_game

קופץ ל player\_lose

**player\_win:**

קורא ל newline

קורא ל newline

השם ב dx את מצביע ל msg\_show\_word

מכניס את dx לראש המחסנית

קורא ל print\_string

השם ב dx את מצביע ל selectedWord

מכניס את dx לראש המחסנית

קורא ל print\_string

קורא ל newline

קורא ל newline

השם ב dx את מצביע ל msg\_win

מכניס את dx לראש המחסנית

קורא ל print\_string

קורא ל newline

קופץ ל skip\_lose

**player\_lose:**

קורא ל newline

קורא ל newline

השם ב dx את מצביע ל msg\_show\_word

מכניס את dx לראש המחסנית

קורא ל print\_string

השם ב dx את מצביע ל selectedWord

מכניס את dx לראש המחסנית

קורא ל print\_string

קורא ל newline

קורא ל newline

השם ב dx את מצביע ל msg\_lose

מכניס את dx לראש המחסנית

קורא ל print\_string

קורא ל newline

**skip\_lose:**

השם ב dx את מצביע ל msg\_Good\_Bye

מכניס את dx לראש המחסנית

קורא ל print\_string

**סוף**

**------------------------------------------------------------------------------------**

**תחילת תת תוכנית 1 (menu)**

מכניס את כל האוגרים לראש המחסנית

השם ב dx את מצביע ל Logo

מכניס את dx לראש המחסנית

קורא ל print\_string

קורא ל newline

השם ב dx את מצביע ל msg\_hangman\_info

מכניס את dx לראש המחסנית

קורא ל print\_string

מחסיר 2 מ sp

מכניס את bp לראש המחסנית

מכניס את הערך 20 לראש המחסנית

קורא ל generate\_random\_number

השם ב bp את sp

השם ב dx את [bp+2] במחסנית

השם ב bp את הערך בראש המחסנית

מוסיף ל sp את הערך 2

השם ב randomNum את dx

מחסיר 2 מ sp

מכניס את bp לראש המחסנית

מכניס את הערך 3 לראש המחסנית

קורא ל generate\_random\_number

השם ב bp את sp

השם ב ax את [bp+2] במחסנית

השם ב bp את הערך בראש המחסנית

מוסיף ל sp את הערך 2

השם ב si את ax

מכפיל ב 2 את si

השם ב dx את הכתובת של [topics+si] ממקטע הזיכרון

מכניס את dx לראש המחסנית

קורא ל print\_string

קורא ל newline

השם ב bx ערך 20

הכפל ax ב bx

הוסף ל randomNum את ax

השם ב si את randomNum

הכפל si ב 2

השם ב dx מבציע ל [words + si] ממקטע הזיכרון

השם ב selectedWord את dx

מוציא את כל האוגרים מראש המחסנית

קופץ לכתובת שבראש המחסנית

**סוף תת תוכנית 1 (menu)**

**------------------------------------------------------------------------------------**

**תחילת תת תוכנית 2 (print\_string)**

מכניס את כל האוגרים לראש המחסנית

השם ב bp את sp

השם ב dx ערך [bp+18] מהמחסנית

השם ב ah את הערך 9

קורא לפונקציה במיקום 21h

מוציא כל האורים מראש המחסנית

הוסף 2 ל sp

קופץ לכתובת שבראש המחסנית

**סוף תת תוכנית 2 (print\_string)**

**------------------------------------------------------------------------------------**

**תחילת תת תוכנית 3 (updateHiddenWord)**

מכניס את כל האוגרים לראש המחסנית

השם ב si את המצביע ל selectedWord

השם ב bx את המצביע ל old\_letters

**loop\_process\_word:**

השם ב al את הערך שמצביע עליו si

ב di את הערך שמצביע עליו old\_letters

**check\_guessed\_letters:**

אם הערך שבכתובת di שווה ל 0:

קפוץ ל add\_underscore

השם ב ah את הערך שמצביע עליו di

אם al == ah:

קופץ ל add\_char\_letter

מוסיף ל di אחד

קופץ ל check\_guessed\_letters

**add\_char\_letter:**

השם ב ah את הערך שמצביע עליו bx

קפוץ ל continue\_processing

**add\_underscore:**

השם בערך שמצביע עליו bx את '\_'

**continue\_processing:**

הוסף ל si אחד

הוסף ל bx אחד

אם הערך שבכתובת si לא שווה ל '$' :

קפוץ ל loop\_process\_word

השם בערך שמצביע bx את '$'

מוציא את כל האוגרים מראש המחסנית

קופץ לכתובת שבראש המחסנית

**סוף תת תוכנית 3 (updateHiddenWord)**

**------------------------------------------------------------------------------------**

**תחילת תת תוכנית 4 (check\_win)**

מכניס את si לראש המחסנית

מכניס את ax לראש המחסנית

השם ב si את המצביע של hidden\_word

השם ב dl את הערך 1

**check\_win\_loop:**

השם ב al את הערך שמצביע עליו si

אם al לא שווה ל '\_':

קפוץ ל skip\_check\_win\_loop

השם ב dl את הערך 0

**skip\_check\_win\_loop:**

הוסף ל si אחד

אם al לא שווה ל '$':

קופץ ל check\_win\_loop

השם ב [bp+8] במחסנית את הערך dx

מוציא את הערך שבראש המחסנית ל ax

מוציא את הערך שבראש המחסנית ל si

קופץ לכתובת שבראש המחסנית

**סוף תת תוכנית 4 (check\_win)**

**------------------------------------------------------------------------------------**

**תחילת תת תוכנית 5 (check\_correct\_guess)**

מכניס את כל האוגרים לראש המחסנית

השם ב si את המצביע ל hidden\_word

השם ב cl את הערך 1-

השם ב dl את הערך 0

**proces\_word:**

השם ב al את הערך שמצביע עליו si

אם al שווה ל '\_':

קפוץ ל add\_num\_hidden\_letter

מוסף ל cl אחד

**add\_num\_hidden\_letter:**

הוסף ל si אחד

אם al לא שווה ל '$'

קופץ ל proces\_word

משווה את cl ל num\_hidden\_letter

אם cl קטן או שווה מ num\_hidden\_letter:

קפוץ ל skip\_hidden\_letter

השם ב num\_hidden\_letter את הערך cl

השם ב dl את הערך 1

השם ב cx את המצביע ל msg\_hidden\_guess

מכניס את cx לראש המחסנית

קורא ל msg\_hidden\_guess

קורא ל newline

**skip\_hidden\_letter:**

השם ב bp את sp

השם ב [bp + 20] במחסנית את dx

מוציא את כל האוגרים מראש המחסנית

קופץ לכתובת שבראש המחסנית

**סוף תת תוכנית 5 (check\_correct\_guess)**

**------------------------------------------------------------------------------------**

**תחילת תת תוכנית 6 (get\_input)**

מכניס את כל האוגרים לראש המחסנית

השם ב dx את מצביע ל msg\_get\_char\_input

מכניס את dx לראש המחסנית

קורא ל print\_string

מחסיר 2 מ sp

מכניס את bp לראש המחסנית

קורא ל get\_char

השם ב bp את sp

השם ב ax את [bp+2] במחסנית

השם ב bp את הערך בראש המחסנית

מוסיף ל sp את הערך 2

מכניס את ax לראש המחסנית

מחסיר 2 מ sp

מכניס את bp לראש המחסנית

מכניס את ax לראש המחסנית

קורא ל check\_illegal\_letter

השם ב bp את sp

השם ב bx את [bp+2] במחסנית

השם ב bp את הערך בראש המחסנית

מוסיף ל sp את הערך 2

מושך את ax מראש המחסנית

אם bl != 1:

קופץ ל skip\_check\_already\_selected

מחסיר 2 מ sp

מכניס את bp לראש המחסנית

מכניס את ax לראש המחסנית

קורא ל check\_illegal\_letter

השם ב bp את sp

השם ב bx את [bp+2] במחסנית

השם ב bp את הערך בראש המחסנית

מוסיף ל sp את הערך 2

אם bl != 0:

קפוץ ל skip\_print

השם ב dx את מצביע ל msg\_letter\_already\_selected

מכניס את dx לראש המחסנית

קורא ל print\_string

קפוץ ל skip\_print

**skip\_check\_already\_selected:**

השם ב dx את מצביע ל msg\_not\_in\_range

מכניס את dx לראש המחסנית

קורא ל print\_string

**skip\_print:**

השם ב bp את sp

השם ב [bp + 20] במחסנית את bl

מוציא את כל האוגרים מראש המחסנית

קופץ לכתובת שבראש המחסנית

**סוף תת תוכנית 6 (get\_input)**

**------------------------------------------------------------------------------------**

**תחילת תת תוכנית 7 (get\_char)**

מכניס את ax לראש המחסנית

מכניס את bx לראש המחסנית

מכניס את sp לראש המחסנית

השם ב ah את הערך 1

קורא לפונקציה במיקום 21h

קורא newline

השם ב bp את sp

השם ב [bp+10] במחסנית את הערך ax

מוציא את הערך שבראש המחסנית ל sp  
מוציא את הערך שבראש המחסנית ל bx

מוציא את הערך שבראש המחסנית ל ax

קופץ לכתובת שבראש המחסנית

**סוף תת תוכנית 7 (get\_char)**

**------------------------------------------------------------------------------------**

**תחילת תת תוכנית 8 (check\_illegal\_letter)**

מכניס את כל האוגרים לראש המחסנית

השם ב bp את sp

השם ב al ערך [bp+18] מהמחסנית

השם ב cx את הערך 0

אם al < 97:

קופץ ל latter\_not\_in\_range

אם al > 122:

קופץ ל latter\_not\_in\_range

השם ב cl את הערך 1

**latter\_not\_in\_range:**

השם ב [bp+22] במחסנית את cx

מוציא את כל האוגרים מראש המחסנית

הוסף 2 ל sp

קופץ לכתובת שבראש המחסנית

**סוף תת תוכנית 8 (check\_illegal\_letter)**

**------------------------------------------------------------------------------------**

**תחילת תת תוכנית 9 (check\_letter\_exists)**

מכניס את כל האוגרים לראש המחסנית

השם ב bp את sp

השם ב dx ערך [bp+18] מהמחסנית

השם ב si את המצביע ל old\_letters

**check\_loop:**

אם הערך שבכתובת si שווה ל 0:

קפוץ ל letter\_not\_found

אם al שווה לערך שבכתובת si:

קפוץ ל letter\_found

מוסיף ל si אחד

קפוץ ל check\_loop

**letter\_found:**

השם ב bx את הערך 0

קפוץ ל skip2

**letter\_not\_found:**

מכניס את ax לראש המחסנית

קורא ל add\_letter

השם ב bx את הערך 1

**skip2:**

השם ב [bp+22] במחסנית את bx

מוציא את כל האוגרים מראש המחסנית

הוסף 2 ל sp

קופץ לכתובת שבראש המחסנית

**סוף תת תוכנית 9 (check\_letter\_exists)**

**------------------------------------------------------------------------------------**

**תחילת תת תוכנית 10 (add\_letter)**

מכניס את כל האוגרים לראש המחסנית

השם ב bp את sp

השם ב dx ערך [bp+18] מהמחסנית

השם ב cx את הערך 0

השם ב si את מצביע ל old\_letters

**check\_add\_letter\_loop:**

אם הערך שבכתובת si שווה ל 0:

קפוץ ל skip\_check\_add\_letter\_loop

הוסף ל si אחד

הוסף ל cx אחד

קפוץ ל check\_add\_letter\_loop

**skip\_check\_add\_letter\_loop:**

אם cx שווה לקבוע MAX\_LETTERS:

קפוץ ל letter\_arr\_full

השם bx את המצביע ל old\_letters

הוסף bx ל cx

השם בערך bx מצביע עליו את al

**letter\_arr\_full:**

מוציא את כל האוגרים מראש המחסנית

הוסף 2 ל sp

קופץ לכתובת שבראש המחסנית

**סוף תת תוכנית 10 (add\_letter)**

**------------------------------------------------------------------------------------**

**תחילת תת תוכנית 11 (print\_letter\_guses)**

מכניס את כל האוגרים לראש המחסנית

השם ב bx את המצביע ל old\_letters

**loop\_old\_letters:**

השם ב ch את הערך ש bx מצביע עליו

השם ב cl את הערך 1

השם ב si את המצביע של selectedWord

**loop\_selectedWord:**

השם ב al את הערך ש si מצביע עליו

מוסיף ל si אחד

אם ch לא שווה ל al:

קופץ ל skip\_change\_cl

השם ב cl את הערך 0

**skip\_change\_cl:**

אם al לא שווה ל '$'

קופץ ל loop\_selectedWord

אם cl != 1:

קופץ ל print\_char

השם ב dl את ch

השם ב ah את 2

קורא לפונקציה במיקום 21h

השם ב dl את הערך '\_'

קורא לפונקציה במיקום 21h

**print\_char:**

הוסף ל bx אחד

אם ch !=0

קופץ ל loop\_old\_letters

מוציא את כל האוגרים מראש המחסנית

קופץ לכתובת שבראש המחסנית

**סוף תת תוכנית 11 (print\_letter\_guses)**

**------------------------------------------------------------------------------------**

**תחילת תת תוכנית 12 (generate\_random\_number)**

מכניס את כל האוגרים לראש המחסנית

השם ב ah את הערך 02Ch

השם ב ah את הערך 0

קורא לפונקציה במיקום 1ah

השם ב bp את sp

השם ב ax את dx

השם ב dx את הערך 0

השם ב dx ערך [bp+18] מהמחסנית

לחלק ax ב bx

השם ב bp את sp

השם ב [bp+22]מהמחסנית את הערך dx

מוציא את כל האוגרים מראש המחסנית

הוסף 2 ל sp

קופץ לכתובת שבראש המחסנית

**סוף תת תוכנית 12 (generate\_random\_number)**

**------------------------------------------------------------------------------------**

**תחילת תת תוכנית 13 (newline)**

מכניס את ax לראש המחסנית

מכניס את dx לראש המחסנית

השם ב ah את הערך 02h

השם ב dl את הערך 10

קורא לפונקציה במיקום 21h

השם ב dl את הערך 13

קורא לפונקציה במיקום 21h

מוציא את dx לראש המחסנית

מוציא את ax לראש המחסנית

קופץ לכתובת שבראש המחסנית

**סוף תת תוכנית 13 (newline)**

**------------------------------------------------------------------------------------**

**סוף פרויקט**

**יישום:**

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

; Constants for the game

MAX\_LETTERS equ 26 ; the amount of letters in English

MAX\_LETTERS\_IN\_WORD equ 8 ; the max amount of letters in word is 8 with $

ROUNDS equ 7 ; number of rounds

.model small

.stack 100h

.data

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

; The Logo

Logo:

db " \_ \_ ", 10,13

db " | | | | ", 10,13

db " | |\_\_| | \_\_ \_ \_ \_\_ \_\_ \_ \_ \_\_ \_\_\_ \_\_ \_ \_ \_\_ ", 10,13

db " | \_\_ |/ \_` | '\_ \ / \_` | '\_ ` \_ \ / \_` | '\_ \ ", 10,13

db " | | | | (\_| | | | | (\_| | | | | | | (\_| | | | |", 10,13

db " |\_| |\_|\\_\_,\_|\_| |\_|\\_\_, |\_| |\_| |\_|\\_\_,\_|\_| |\_|", 10,13

db " \_\_/ | ", 10,13

db " |\_\_\_/", 10,13

db "$"

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

; Info about the game

msg\_hangman\_info:

db "Welcome to Hangman!", 10,13

db "Guess the hidden word by selecting one letter at a time.", 10,13

db "Each incorrect guess adds a part to the hangman figure.", 10,13

db "You have a limited number of attempts to guess correctly.", 10,13

db "Good luck and have fun!", 10,13

db "", 10,13

db "The theme is: ", 10,13

db "$"

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

; End messages

msg\_win:

db " \_\_\_\_ \_\_\_\_ \_\_\_\_\_ \_\_\_\_ \_\_\_\_\_ \_\_\_\_ \_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ ", 10,13

db "|\_ \_| |\_ \_||\_ \_||\_ \|\_ \_||\_ \|\_ \_||\_ \_\_ ||\_ \_\_ \ ", 10,13

db " \ \ /\ / / | | | \ | | | \ | | | |\_ \\_| | |\_\_) | ", 10,13

db " \ \/ \/ / | | | |\ \| | | |\ \| | | \_| \_ | \_\_ / ", 10,13

db " \ /\ / \_| |\_ \_| |\_\ |\_ \_| |\_\ |\_ \_| |\_\_/ | \_| | \ \\_ ", 10,13

db " \/ \/ |\_\_\_\_\_||\_\_\_\_\_|\\_\_\_\_||\_\_\_\_\_|\\_\_\_\_||\_\_\_\_\_\_\_\_||\_\_\_\_| |\_\_\_| ", 10,13

db " ", 10,13

db " Y O U W I N :) ", 10,13

db "$"

msg\_lose:

db " \_ \_\_\_\_ \_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_ ", 10,13

db " | | / \_\_ \ / \_\_\_\_| \_\_\_\_|| \_\_ \", 10,13

db " | | | | | | (\_\_\_ | |\_\_ || |\_\_) |", 10,13

db " | | | | | |\\_\_\_ \| \_\_| | \_ / ", 10,13

db " | |\_\_\_| |\_\_| |\_\_\_\_) | |\_\_\_\_ | | \ \ ", 10,13

db " |\_\_\_\_\_\_\\_\_\_\_/|\_\_\_\_\_/|\_\_\_\_\_\_||\_| \\_\", 10,13

db " ", 10,13

db " Y O U L O S E ): ", 10,13

db " ", 10,13

db "$"

msg\_Good\_Bye:

db " \_\_\_\_\_ \_ \_\_\_\_ \_ ", 10,13

db " / \_\_\_\_| | | | \_ \ | |", 10,13

db " | | \_\_ \_\_\_ \_\_\_ \_\_| | | |\_) |\_ \_ \_\_\_| |", 10,13

db " | | |\_ |/ \_ \ / \_ \ / \_` | | \_ <| | | |/ \_ \ |", 10,13

db " | |\_\_| | (\_) | (\_) | (\_| | | |\_) | |\_| | \_\_/\_|", 10,13

db " \\_\_\_\_\_|\\_\_\_/ \\_\_\_/ \\_\_,\_| |\_\_\_\_/ \\_\_, |\\_\_\_(\_)", 10,13

db " \_\_/ | ", 10,13

db " |\_\_\_/ ", 10,13

db " ", 10,13

db " GOOD BYE! SEE YOU NEXT TIME ", 10,13

db "$"

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

; All the states of the Hanged man

Hanged\_man0:

db " \_\_\_\_\_\_\_", 10,13

db " | |", 10,13

db " | ", 10,13

db " | ", 10,13

db " | ", 10,13

db " | ", 10,13

db "\_\_|\_\_ $", 10,13

Hanged\_man1:

db " \_\_\_\_\_\_\_", 10,13

db " | |", 10,13

db " | O", 10,13

db " | ", 10,13

db " | ", 10,13

db " | ", 10,13

db "\_\_|\_\_ $", 10,13

Hanged\_man2:

db " \_\_\_\_\_\_\_", 10,13

db " | |", 10,13

db " | O", 10,13

db " | |", 10,13

db " | ", 10,13

db " | ", 10,13

db "\_\_|\_\_ $", 10,13

Hanged\_man3:

db " \_\_\_\_\_\_\_", 10,13

db " | |", 10,13

db " | O", 10,13

db " | /|", 10,13

db " | ", 10,13

db " | ", 10,1

db "\_\_|\_\_ $", 10,13

Hanged\_man4:

db " \_\_\_\_\_\_\_", 10,13

db " | |", 10,13

db " | O", 10,13

db " | /|\", 10,13

db " | ", 10,13

db " | ", 10,13

db "\_\_|\_\_ $", 10,13

Hanged\_man5:

db " \_\_\_\_\_\_\_", 10,13

db " | |", 10,13

db " | O", 10,13

db " | /|\", 10,13

db " | / ", 10,13

db " | ", 10,13

db "\_\_|\_\_ $", 10,13

Hanged\_man6:

db " \_\_\_\_\_\_\_", 10,13

db " | |", 10,13

db " | O", 10,13

db " | /|\", 10,13

db " | / \", 10,13

db " | ", 10,13

db "\_\_|\_\_ $", 10,13

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

; Array of pointers with all the Hanged\_mans

Hanged\_mans:

dw offset Hanged\_man0, offset Hanged\_man1, offset Hanged\_man2

dw offset Hanged\_man3, offset Hanged\_man4, offset Hanged\_man5

dw offset Hanged\_man6

Hanged\_man\_index: dw 0

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

;The words in the game

; Fruits

fruits0 db "apple$"

fruits1 db "banana$"

fruits2 db "grape$"

fruits3 db "orange$"

fruits4 db "melon$"

fruits5 db "cherry$"

fruits6 db "peach$"

fruits7 db "kiwi$"

fruits8 db "plum$"

fruits9 db "pear$"

fruits10 db "lemon$"

fruits11 db "fig$"

fruits12 db "papaya$"

fruits13 db "mango$"

fruits14 db "guava$"

fruits15 db "berry$"

fruits16 db "date$"

fruits17 db "olive$"

fruits18 db "apricot$"

fruits19 db "coconut$"

; Animals

animals0 db "cat$"

animals1 db "dog$"

animals2 db "lion$"

animals3 db "bear$"

animals4 db "wolf$"

animals5 db "fox$"

animals6 db "deer$"

animals7 db "zebra$"

animals8 db "horse$"

animals9 db "sheep$"

animals10 db "goat$"

animals11 db "mouse$"

animals12 db "shark$"

animals13 db "whale$"

animals14 db "snake$"

animals15 db "eagle$"

animals16 db "tiger$"

animals17 db "rabbit$"

animals18 db "camel$"

animals19 db "otter$"

; Cities

cities0 db "paris$"

cities1 db "london$"

cities2 db "tokyo$"

cities3 db "berlin$"

cities4 db "madrid$"

cities5 db "rome$"

cities6 db "sydney$"

cities7 db "delhi$"

cities8 db "moscow$"

cities9 db "dublin$"

cities10 db "vienna$"

cities11 db "cairo$"

cities12 db "oslo$"

cities13 db "lisbon$"

cities14 db "sofia$"

cities15 db "seoul$"

cities16 db "beirut$"

cities17 db "athens$"

cities18 db "baghdad$"

cities19 db "doha$"

;Array of pointers with all the words

words dw offset fruits0, offset fruits1, offset fruits2, offset fruits3, offset fruits4

dw offset fruits5, offset fruits6, offset fruits7, offset fruits8, offset fruits9

dw offset fruits10, offset fruits11, offset fruits12, offset fruits13, offset fruits14

dw offset fruits15, offset fruits16, offset fruits17, offset fruits18, offset fruits19

dw offset animals0, offset animals1, offset animals2, offset animals3, offset animals4

dw offset animals5, offset animals6, offset animals7, offset animals8, offset animals9

dw offset animals10, offset animals11, offset animals12, offset animals13, offset animals14

dw offset animals15, offset animals16, offset animals17, offset animals18, offset animals19

dw offset cities0, offset cities1, offset cities2, offset cities3, offset cities4

dw offset cities5, offset cities6, offset cities7, offset cities8, offset cities9

dw offset cities10, offset cities11, offset cities12, offset cities13, offset cities14

dw offset cities15, offset cities16, offset cities17, offset cities18, offset cities19

;Topics array

topic0 db "Fruits!$",0

topic1 db "Animals!$",0

topic2 db "Cities!$",0

; Array of pointers to the topics

topics dw offset topic0, offset topic1, offset topic2

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

;msg

msg\_get\_char\_input db "Enter a letter: $"

msg\_letter\_already\_selected db "Letter already chosen, $"

msg\_not\_in\_range db "Letter not in range a-z, $"

msg\_hidden\_guess db 'You guessed a correct letter!$'

msg\_incorrect\_guess db "Wrong guess! Better luck next time!$"

msg\_round db "Round: $"

msg\_wrong\_guesses db "Wrong Guesses: $"

msg\_show\_word db "The word is: $"

msg\_another\_game db "Do you want to play another game? (Y/N): $"

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

;Variables

randomNum dw 0 ;varibale for random number

old\_letters db MAX\_LETTERS dup(0) ;26 cells with valuse of 0

selectedWord dw ? ;the random word

hidden\_word db MAX\_LETTERS\_IN\_WORD dup(?);the word with the higher char it strawberry$ with 11

num\_hidden\_letter db 0

.code

mov ax,@data

mov ds, ax

call menu;call menu function

call updateHiddenWord

mov cx,ROUNDS

main\_game:

mov si,Hanged\_man\_index

shl si,1

courrect\_letter\_loop:

;check win

sub sp,2;Saves space in the stack

push bp;save bp

call check\_win

mov bp,sp

mov dx,[bp+2];pull the value

pop bp;pull bp

add sp,2;Returns the stack to the source

cmp dl,1

je player\_win ;check if the player fill all the ;hidden\_word not have \_

;print rounds

call newline

call newline

mov dx,offset msg\_round;print roundom msg

push dx

call print\_string

mov dl, cl ; Load the digit into dl

add dl, '0' ; Convert the digit to ASCII

mov ah, 2 ; Select DOS function 2 (print character)

int 21h ; Print the character in dl

call newline

;print worng letters

mov dx,offset msg\_wrong\_guesses

push dx

call print\_string

call print\_letter\_guses

call newline

call newline

;print hidden\_word

mov dx,offset hidden\_word

push dx

call print\_string

call newline

call newline

;get char and check input

enter\_agin:

sub sp,2;Saves space in the stack

push bp;save bp

call get\_input

mov bp,sp

mov dx,[bp+2];pull the value

pop bp;pull bp

add sp,2;Returns the stack to the source

cmp dl,1

jne enter\_agin;if dl = 0 run agine

;fill hidden\_word

call updateHiddenWord

;check if the user put courrect letter

sub sp,2;Saves space in the stack

push bp;save bp

call check\_correct\_guess

mov bp,sp

mov dx,[bp+2];pull the value

pop bp;pull bp

add sp,2;Returns the stack to the source

cmp dl,1

je courrect\_letter\_loop;0 worng guse ,

;print msg\_incorrect\_guess

mov dx,offset msg\_incorrect\_guess

push dx

call print\_string

call newline

;print Hanged\_mans

mov dx,[Hanged\_mans + si]

push dx

call print\_string

call newline

inc Hanged\_man\_index

loop main\_game

;if the player come to here he finish the rounds

jmp player\_lose

player\_win:

call newline

call newline

mov dx,offset msg\_show\_word;print the The word is

push dx

call print\_string

mov dx,selectedWord;print the word

push dx

call print\_string

call newline

call newline

mov dx,offset msg\_win;win msg

push dx

call print\_string

call newline

jmp skip\_lose

player\_lose:

call newline

call newline

mov dx,offset msg\_show\_word;print the The word is

push dx

call print\_string

mov dx,selectedWord;print the word

push dx

call print\_string

call newline

call newline

call newline

call newline

mov dx,offset msg\_lose;lose msg

push dx

call print\_string

call newline

skip\_lose:

;good bye msg

mov dx,offset msg\_Good\_Bye

push dx

call print\_string

mov ah,4ch;end of the main

mov al,0 ;end of the main

int 21h ;end of the main

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

; Function: Run in the beggining of the game and, print logo,choose the random word.

; Input: None.

; Output: None.

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

proc menu;Start of the menu function

pusha

;print logo

mov dx, offset Logo; Get the pointer of Logo

push dx;Push to thh stack, for save it

call print\_string

call newline

;print info

mov dx,offset msg\_hangman\_info;the info about the game

push dx

call print\_string

;generat num 0-19

sub sp,2;Saves space in the stack

push bp;save bp

push 20

call generate\_random\_number

mov bp,sp

mov dx,[bp+2];pull the value

pop bp;pull bp

add sp,2;Returns the stack to the source

mov randomNum,dx

;generat num 0-2

sub sp,2;Saves space in the stack

push bp;save bp

push 3

call generate\_random\_number

mov bp,sp

mov ax,[bp+2];pull the value

pop bp;pull bp

add sp,2;Returns the stack to the source

;print topic

mov si,ax

shl si,1;mul in 2

mov dx,[topics+si]

push dx

call print\_string

call newline

;mack the random number point on the correct topic

mov bx,20;mul in 20 ax

mul bx

add randomNum,ax

mov si, randomNum;for choose a word in the arr

shl si, 1; because words - 2byte,randomNum - 1byte

mov dx, [words + si];tack the addres and + index

mov selectedWord,dx;save to the later

popa

ret

endp menu;End of the menu function

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

; Function: Prints a string from the stack using DOS interrupt 21h, ah=09h

; Input: Pointer to string (offset), must be in the end'$'.

; Output: None.

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

proc print\_string

pusha ;save all the registers in the stack

mov bp, sp ;bp work with like bx data segment

mov dx, [bp+18] ;18 - because pusha 16,fun 2

mov ah, 09h;Interrupt of print,use dx

int 21h ;Interrupt of print,use dx

popa ;Pull all the registers from the stack and save them in the registers

retn 2 ;For clean the stack, I writh 2 beacuse push only one parameter for the function

endp print\_string

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

; Function: Fill the hidden\_word with '\_' or with letters from old\_letters

; if the letter in selectedWord

; Input: None.

; Output: Return full hidden\_word with '\_' and chars.

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

proc updateHiddenWord

pusha

mov si,selectedWord ; the random word

mov bx,offset hidden\_word ; the array of guesses

loop\_process\_word:

mov al,[si] ; load the character from selectedWord

mov di,offset old\_letters ; the array of guessed letters

check\_guessed\_letters:

cmp [di],0; until old letter end

je add\_underscore ; go to \_

mov ah,[di]

cmp al,ah ; check if letter matches

je add\_char\_letter ; add letter

inc di

jmp check\_guessed\_letters

add\_char\_letter:

mov [bx],ah

jmp continue\_processing

add\_underscore:

mov [bx],'\_'

continue\_processing:

inc si

inc bx

cmp [si],'$';stop if the last char is $

jne loop\_process\_word

mov [bx],'$';add to the str.length + 1 &

popa

ret

endp updateHiddenWord

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

; Function: Check if hidden\_word not have '\_'

; because, if have the need guess more latter.

; Input: Space in stack.

; Output: If the player win return 1 else 0,save it in the stack.

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

proc check\_win

push si

push ax

;mov si,selectedWord ; the random word

mov si,offset hidden\_word ; the array of guesses

mov dl,1;win

check\_win\_loop:

mov al,[si]

cmp al,'\_'

jne skip\_check\_win\_loop

mov dl,0;not win

skip\_check\_win\_loop:

inc si

cmp al,'$'

jne check\_win\_loop

mov bp,sp

mov [bp+8],dx;save dx in space, 8 because 2 x push - 4 ,fun - 2, parameter - 2

pop ax

pop si

ret

endp check\_win

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

; Function: Check if some letter add to hidden\_word.

; Input: Space in stack.

; Output: Return 1 or 0 with the stack, if 1 correct guse, 0 worng guse.

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

proc check\_correct\_guess

pusha

mov si,offset hidden\_word

mov cl,-1;because it count with $

mov dx,0;uncorrect letter

proces\_word:

mov al,[si]; char char hiding\_word

cmp al,'\_'; chech char == '\_'

je add\_num\_hidden\_letter

inc cl

add\_num\_hidden\_letter:

inc si

cmp al,'$';check end of hidden\_word

jne proces\_word

cmp cl,num\_hidden\_letter;check if correct letter add

jbe skip\_hidden\_letter

mov num\_hidden\_letter,cl

mov dl,1;correct letter

;guess correct letter msg

mov cx,offset msg\_hidden\_guess

push cx

call print\_string

call newline

skip\_hidden\_letter:

mov bp,sp

mov [bp + 20],dx; save in the space,20 because pusha - 16, fun - 2, parameter - 2

popa

ret

endp check\_correct\_guess

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

; Function: Do all the input function in the end return in the stack.

; 1 if courrect else 0.

; Input: From the stack.

; Output: Return to the stack 1 if the legal and courrect, 0 if the illegal.

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

proc get\_input

pusha

mov dx,offset msg\_get\_char\_input

push dx

call print\_string ;print Enter a Char:

sub sp,2;Saves space in the stack

push bp;save bp

call get\_char

mov bp,sp

mov ax,[bp+2];pull the char input value

pop bp;pull bp

add sp,2;Returns the stack to the source

push ax;char input value

sub sp,2;Saves space in the stack

push bp;save bp

push ax

call check\_illegal\_letter

mov bp,sp

mov bx,[bp+2];pull the value

pop bp;pull bp

add sp,2;Returns the stack to the source

pop ax;pull char input value

cmp bl,1;if 1 letter in range

jne skip\_check\_already\_selected;if the letter not in the range jmp

sub sp,2;Saves space in the stack

push bp;save bp

push ax;char input value

call check\_letter\_exists;need add

mov bp,sp

mov bx,[bp+2];pull the boolean value

pop bp;pull bp

add sp,2;Returns the stack to the source

cmp bl,0 ; if the letter in the list

jne skip\_print;jmp if letter not in the list

mov dx,offset msg\_letter\_already\_selected

push dx

call print\_string

jmp skip\_print

skip\_check\_already\_selected:

mov dx,offset msg\_not\_in\_range; print not\_in\_range\_msg

push dx

call print\_string

skip\_print:

mov bp,sp

mov [bp + 20],bl

popa

ret

endp get\_input

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

; Function: Get char from the user and do in the end\n function.

; Input: None.

; Output: Return to the stack the char.

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

proc get\_char

push ax

push bp

push sp

mov ah, 01h ; Character input with echo

int 21h ; Character in AL

call newline

mov bp,sp

mov [bp+10],ax

pop sp

pop bp

pop ax

ret

endp get\_char

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

; Function: Check if the letter in range of a-z

; Input: Save space in the stack.

; Output: Return 1 if the letter in the range else retutn 0.

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

proc check\_illegal\_letter

pusha

mov bp,sp

mov al,[bp+18]

mov cx,0

;a-z

cmp al,97 ; al < 97 - a

jb latter\_not\_in\_range

cmp al,122; al > 122 - z

ja latter\_not\_in\_range

mov cl,1; letter in range

latter\_not\_in\_range:

mov [bp+22],cx

popa

retn 2

endp check\_illegal\_letter

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

; Function: Check if the ltter exists in old\_letters.

; Input: From the stack.

; Output: Return in stack the 1 - letter not in the arr,0 - letter in the arr.

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

proc check\_letter\_exists

pusha

mov bp,sp

mov ax,[bp+18]

mov si,offset old\_letters ;the arr

check\_loop:

cmp [si],0 ;check if we dont have more letters to check

je letter\_not\_found

;use si beacuse si creat for work with arr indexs(Continuous memory)

cmp al,[si] ;check if the letter is in the arr[si]

je letter\_found

inc si;go to the next value

jmp check\_loop

letter\_found:

mov bx,0;letter in the arr(not add letter) - 0

jmp skip2;if letter found bl = 1 and skip the found

letter\_not\_found:

push ax;push to add\_letter

call add\_letter

mov bx,1;letter not in the arr(add letter) - 1

skip2:

mov [bp+22],bx

popa

retn 2

endp check\_letter\_exists

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

; Function: Add letter to old letters,if old letters full (26) not add letter from the stack.

; Input: Space in stack.

; Output: None.

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

proc add\_letter

pusha

mov bp,sp

mov ax,[bp+18]

mov cx,0;amout of letter in the arr

mov si,offset old\_letters

;run until the one after the last letter in the arr

check\_add\_letter\_loop:

cmp [si],0

je skip\_check\_add\_letter\_loop

inc si

inc cx

jmp check\_add\_letter\_loop

skip\_check\_add\_letter\_loop:

cmp cx,MAX\_LETTERS;letter\_index == 26

je letter\_arr\_full ;check if the arr is full (26)

mov bx, offset old\_letters;else,go to the first of the old\_letters arr

add bx,cx; bx point of the index in the

mov [bx],al

letter\_arr\_full:

popa

retn 2

endp add\_letter

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

; Function: Printing all the old letters that are not correct guess.

; Input: None.

; Output: None.

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

proc print\_letter\_guses

pusha

mov bx,offset old\_letters

loop\_old\_letters:

mov ch,[bx]

mov cl,1

mov si,selectedWord

loop\_selectedWord:

mov al,[si]; char char from selectedWord

inc si

cmp ch,al; not need to print

jne skip\_change\_cl

mov cl,0 ;if cl = 0 - char not in the word

skip\_change\_cl:

cmp al,'$'

jne loop\_selectedWord

cmp cl,1;if cl = 1 not need to print the char - char in the word

jne print\_char

mov dl,ch ; Print letter ch

mov ah, 2

int 21h

mov dl,' '; pritn ' ' between the chars

int 21h

print\_char:

inc bx

cmp ch,0;check if we got to the end (0)

jne loop\_old\_letters

popa

ret

endp print\_letter\_guses

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

; Function: Generate number bettwen (0 - number in stack-1) and put in the stack.

; Input: Number to div,space in stack.

; Output: Return into stack the random number.

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

proc generate\_random\_number

pusha

mov ah, 02Ch;Get the Time in format of hours, minutes, seconds, and milliseconds (HH:MM:SS:MS)

mov ah,0 ;interrupts to get the system time

int 1ah ;now clocks ticks will be saved in DX

mov bp, sp

mov ax,dx

mov dx,0 ;clear the DX to zero

mov bx,[bp+18] ;generate between 0 - number-1, 18 - pusha 16+fun 2

div bx ;divide ax by bx

mov bp, sp ;bp work with like bx with data segment

mov [bp+22],dx ;22 - because pusha 16,fun 2,push x 2 - 4

popa

retn 2

endp generate\_random\_number

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

; Function: Printing \n witn 13,10 and interapt.

; Input: None.

; Output: None.

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

proc newline

push ax

push dx

mov ah, 02h ; for print only one char

mov dl, 10 ; Line Feed 10

int 21h

mov dl, 13 ; Carriage Return 13

int 21h

pop dx

pop ax

ret

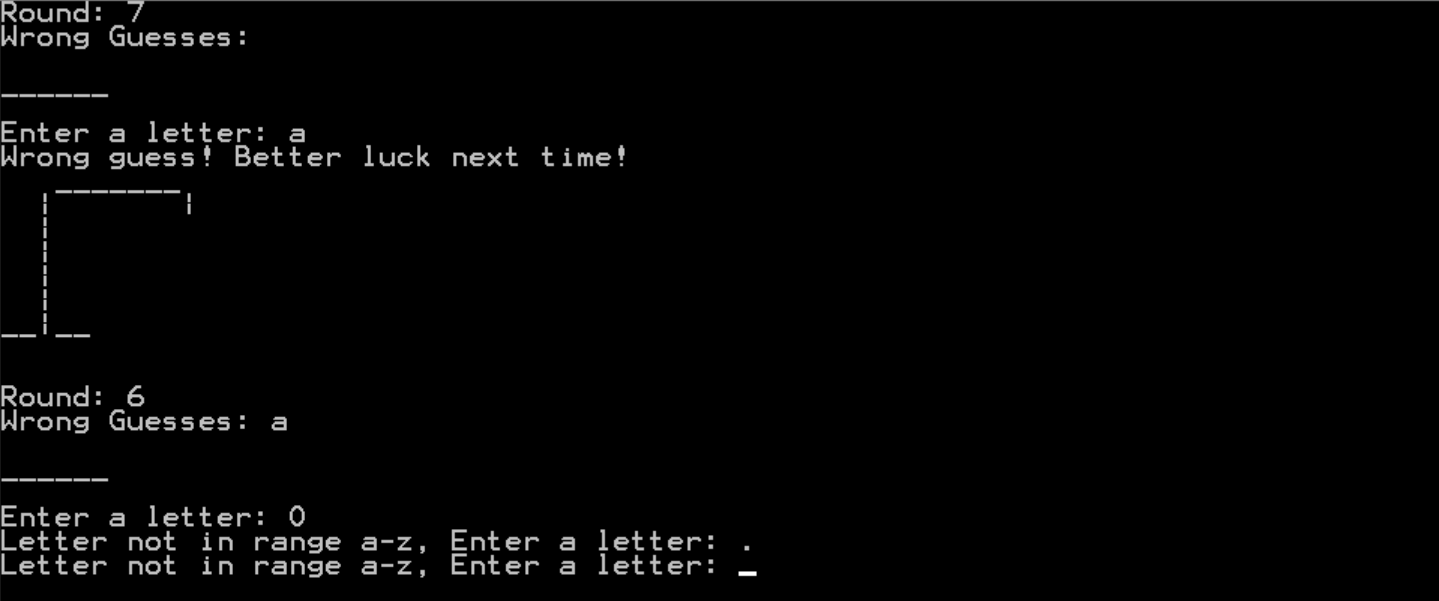
endp newline

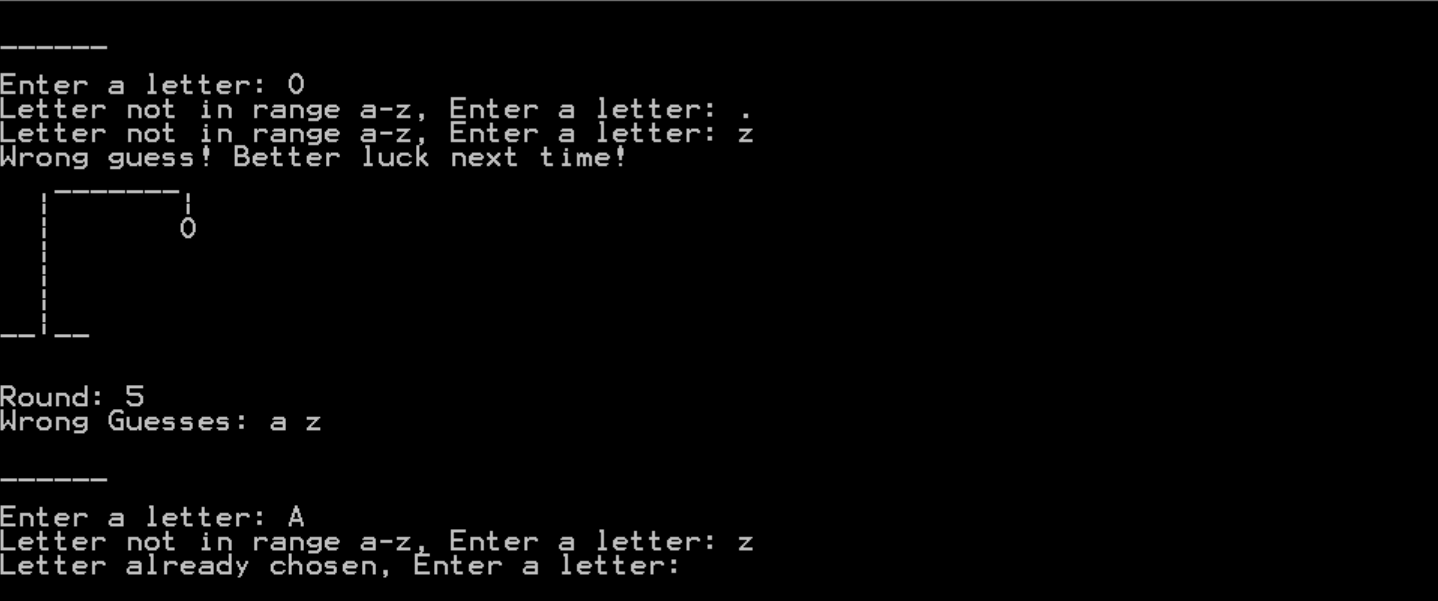
end

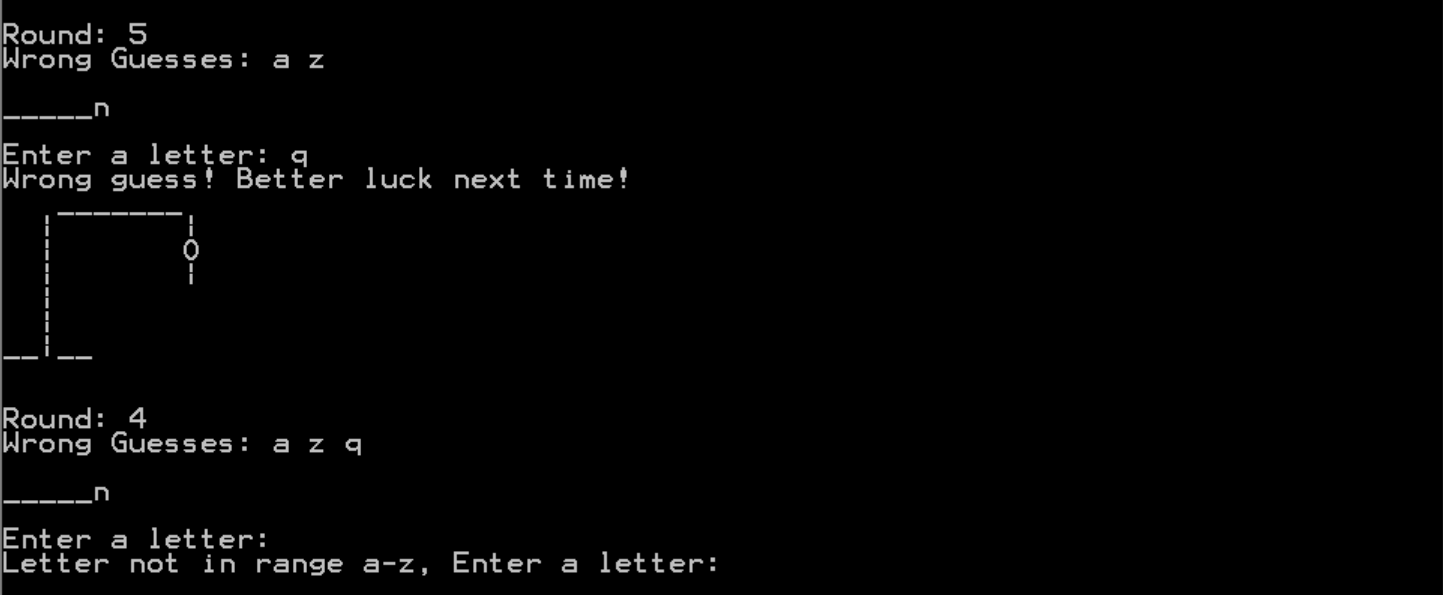
**תמונות מהתוכנית:**

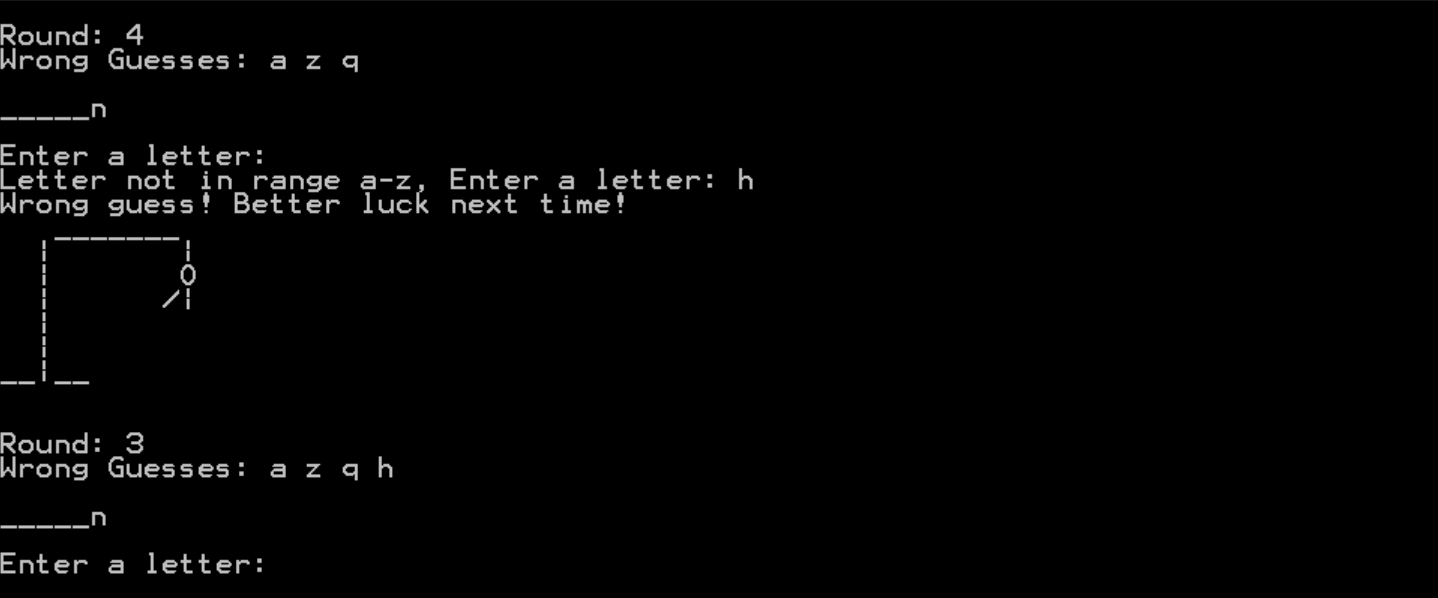
**הפסד:**

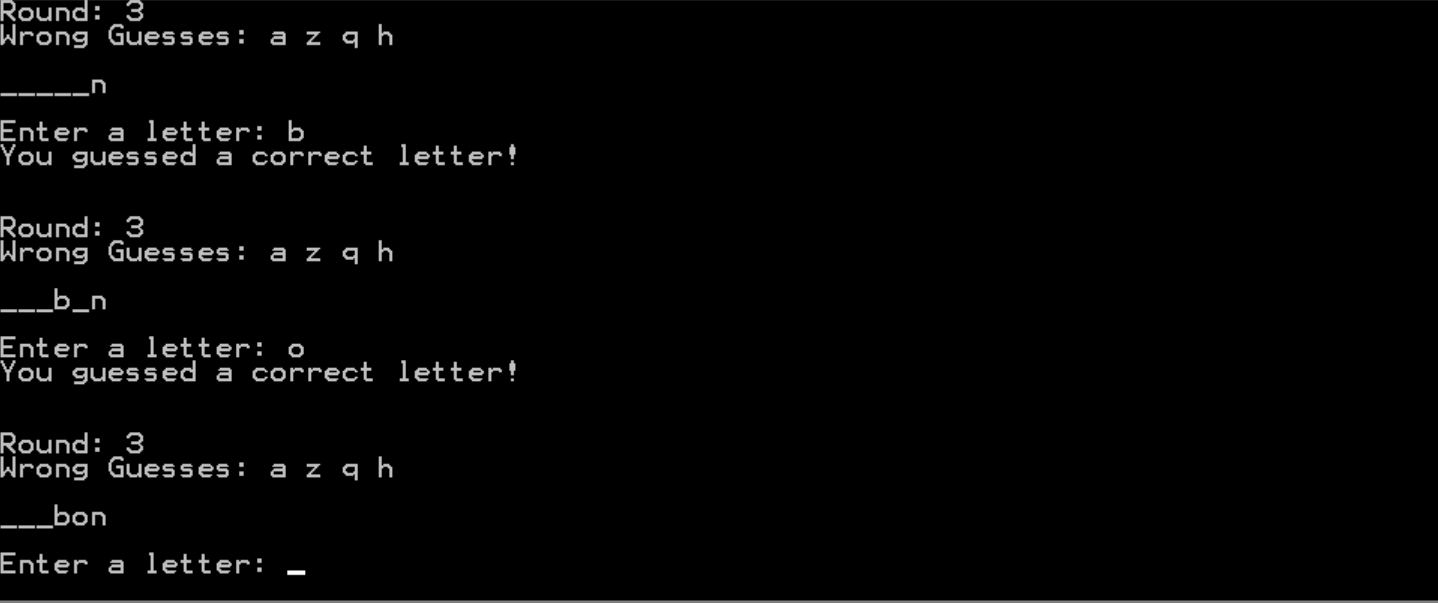
****

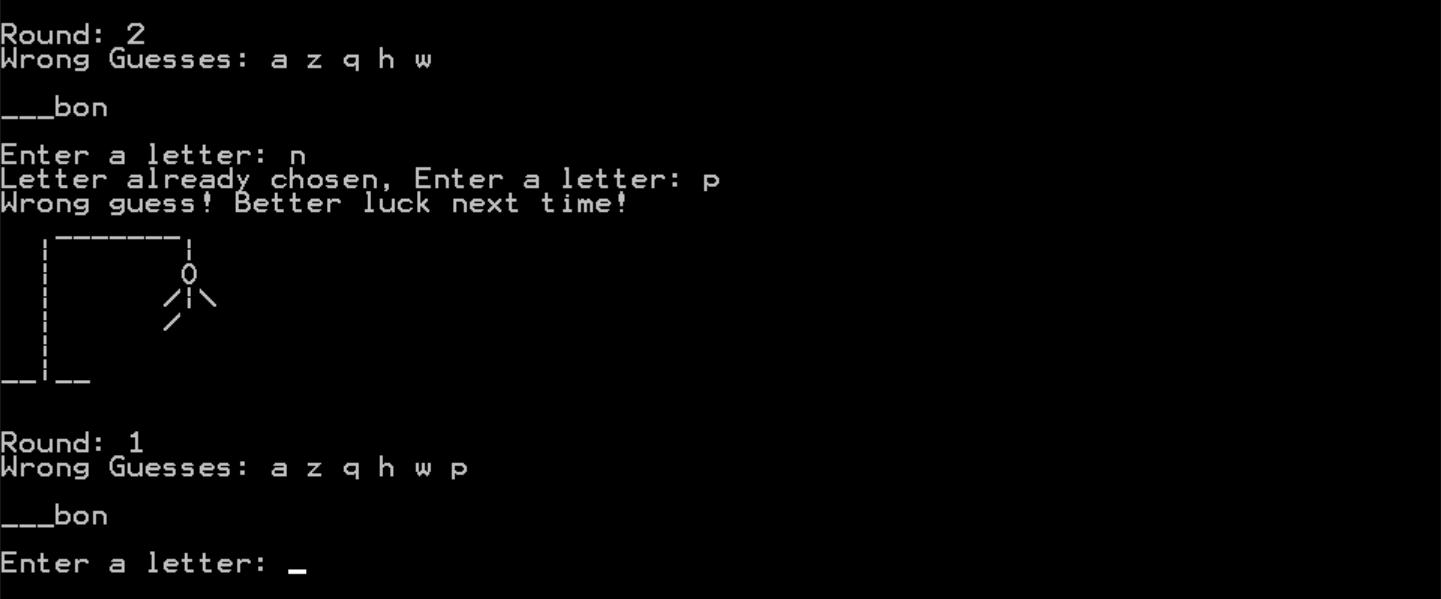
****

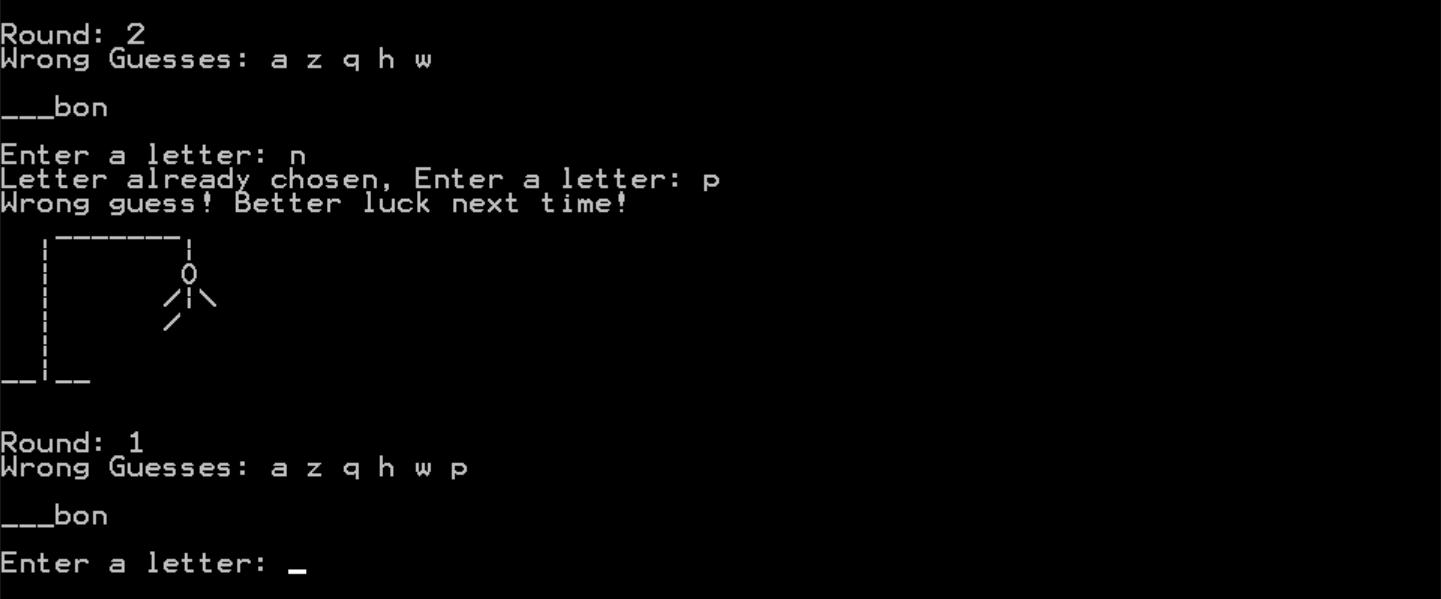
****

****

****

****

****

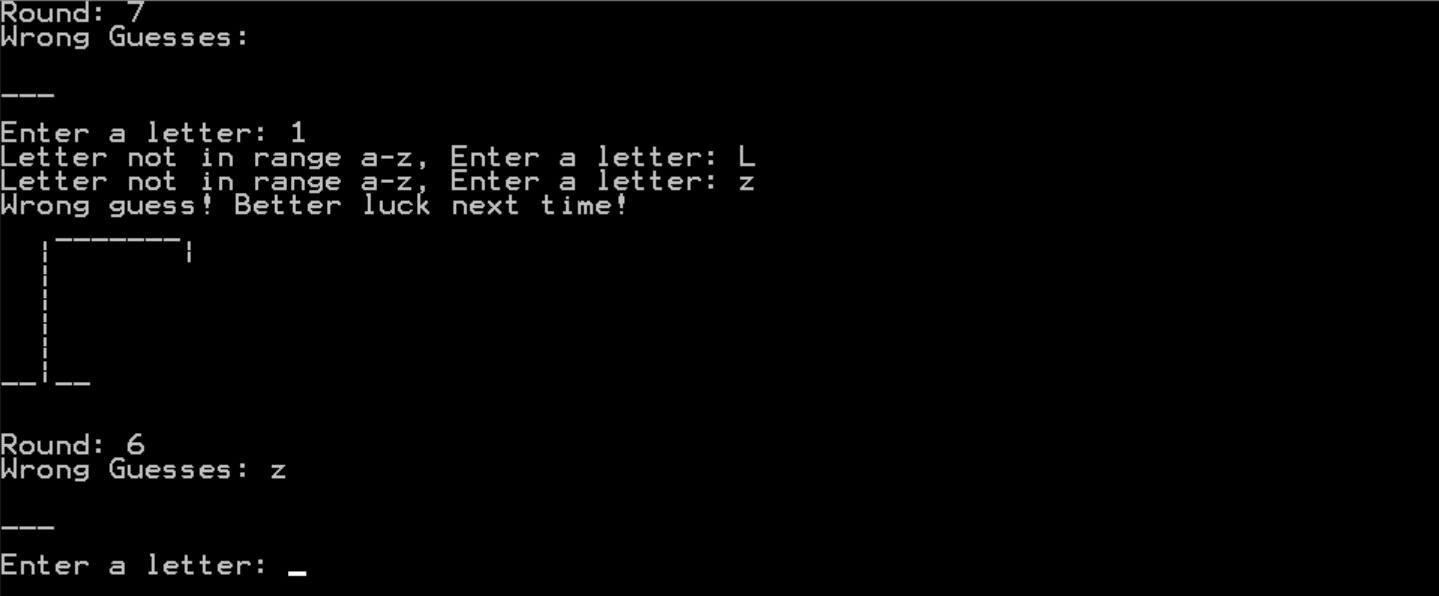
****

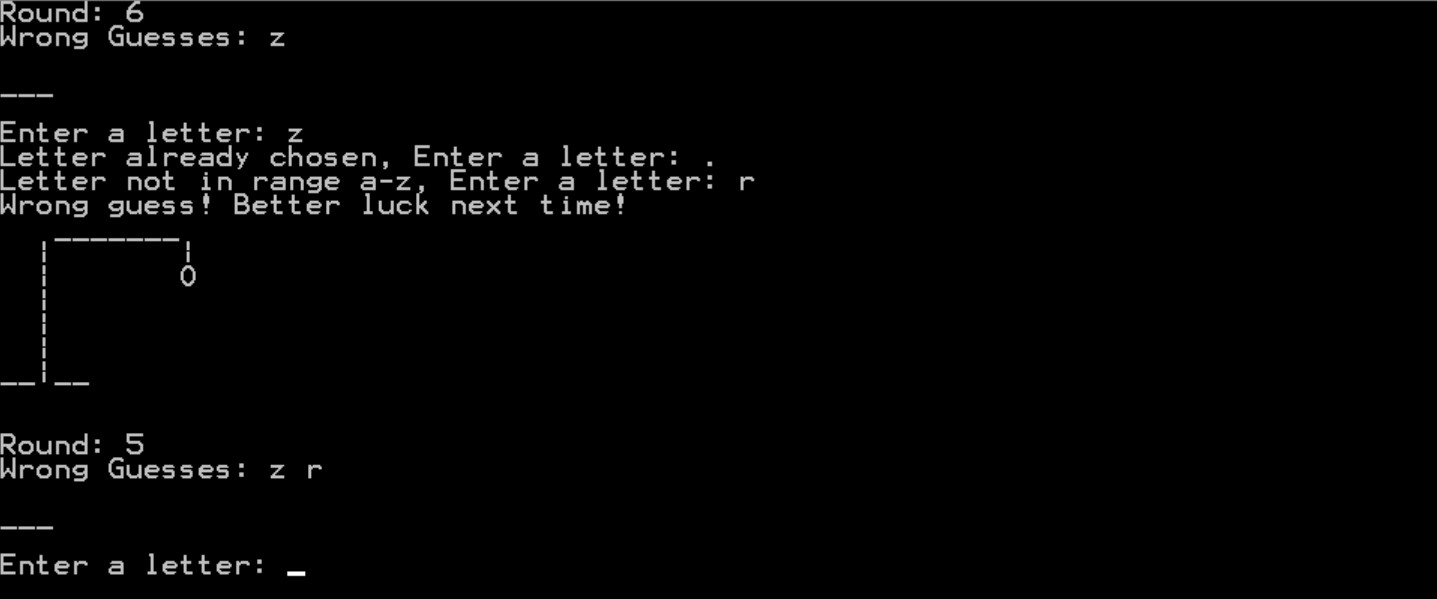
****

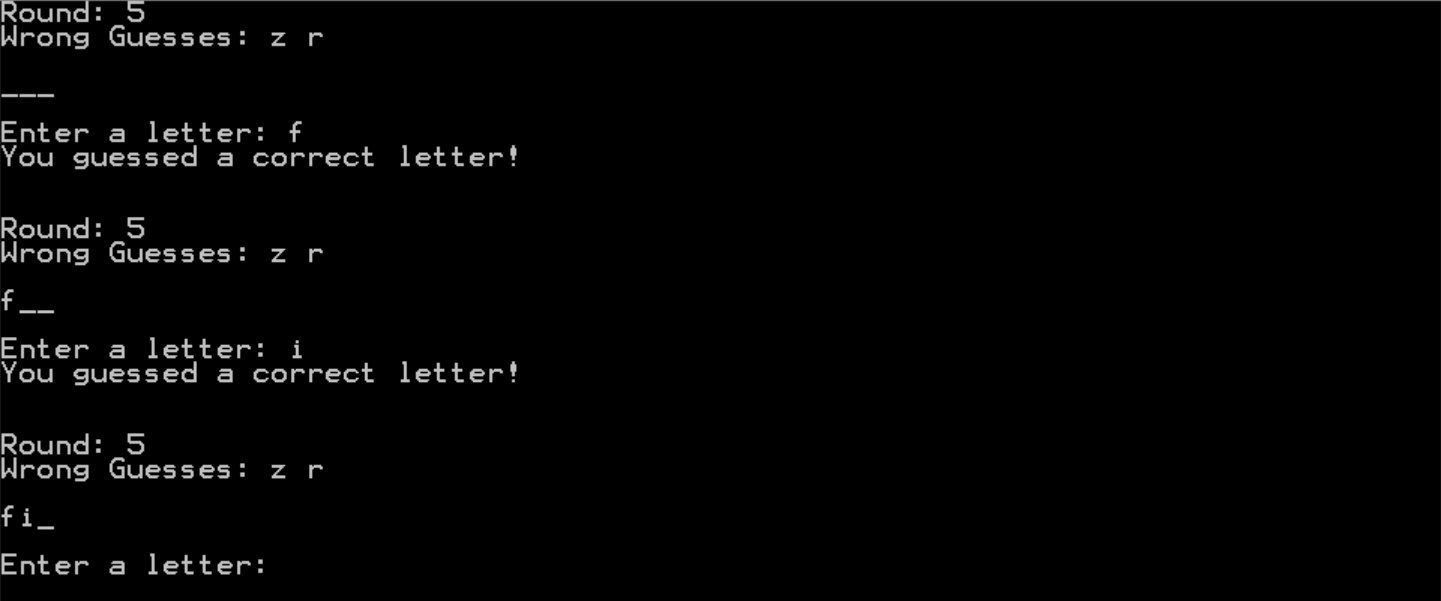
****

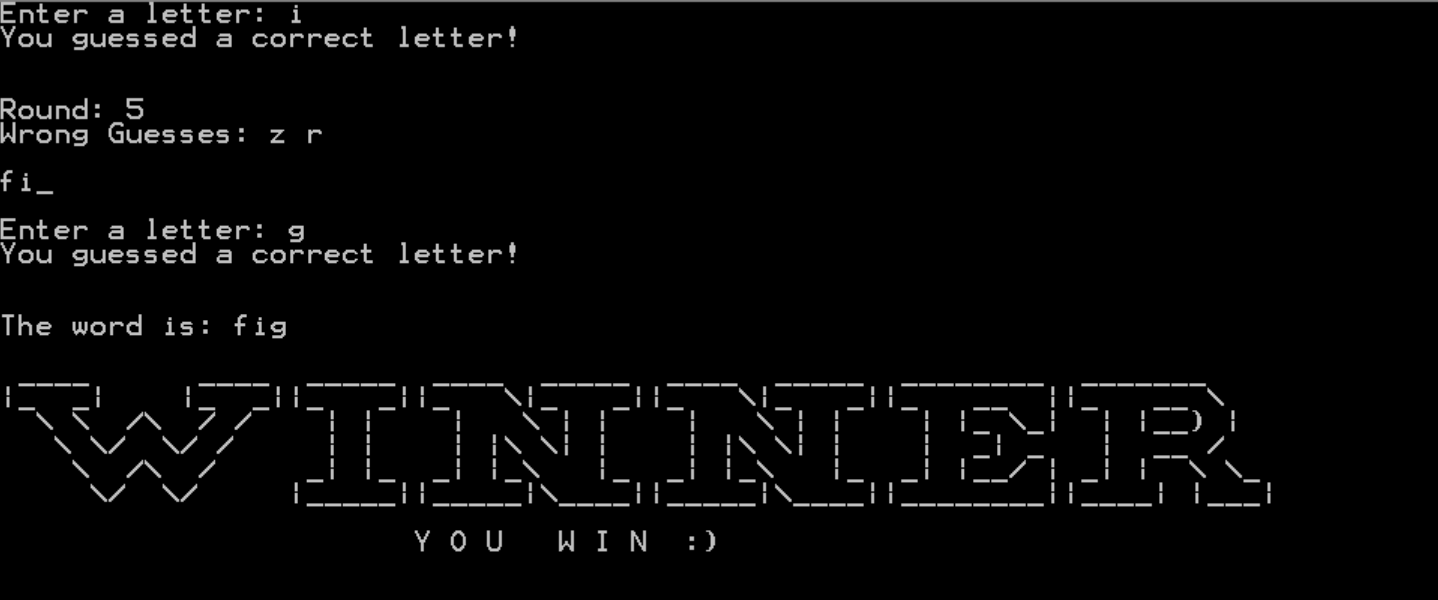
**ניצחון:**

****

****

****

****

****

****