

פרויקט גמר אסמלים

HangMan

שם המשחק : HangMan

מגיש: לייאור שם טוב

כיתה: יא'ג

מורה: איציק בן שטרית

שנה: 2024

בית ספר: מקיף ז'



תוכן עניינים:

המשחק HangMan

עמוד 2

סביבת העבודה

עמודים 4 - 8

הגדרת משתנים

עמוד 9

תת מישימות של הפרויקט

עמוד 10

תתי תוכניות

עמודים 11 - 12

אלגוריתם

עמודים 13 - 28

יישום

עמודים 29 - 58

תמונות מהתוכנית

עמודים 59 - 64

:HangMan

מהות המשחק:

במשחק HangMan, כולן שחקן אחד שמטרתו לנחש מילה נסתרת לפני שמספרים "لتלות" את האיש.

תחילת המשחק:

השחקן מקבל נושא שמרמז על המילה (לדוגמה: "חיות", "מדינות"). המילה מופיעה כמספר קווים תחתונים, שככל אחד מהם מייצג אות חסרה במילה.

מהלך המשחק:

בכל תור השחקן מנחש אות אחת (z-a).

אם האות נכונה, היא נחשפת במקומה במילה.

אם האות שגוייה, האיש התלי מתקרב שלב נוסף לסיומו (נוסף חלק לגוף), ומספר הנסיבות יורד.

המטרה:

לנחש את המילה במלואה לפני שככל חלקו הגוף של האיש התלי מוצגים.

סיום:

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

אם לא, מוצגת המילה והשחקן מסתיים בהפסד.



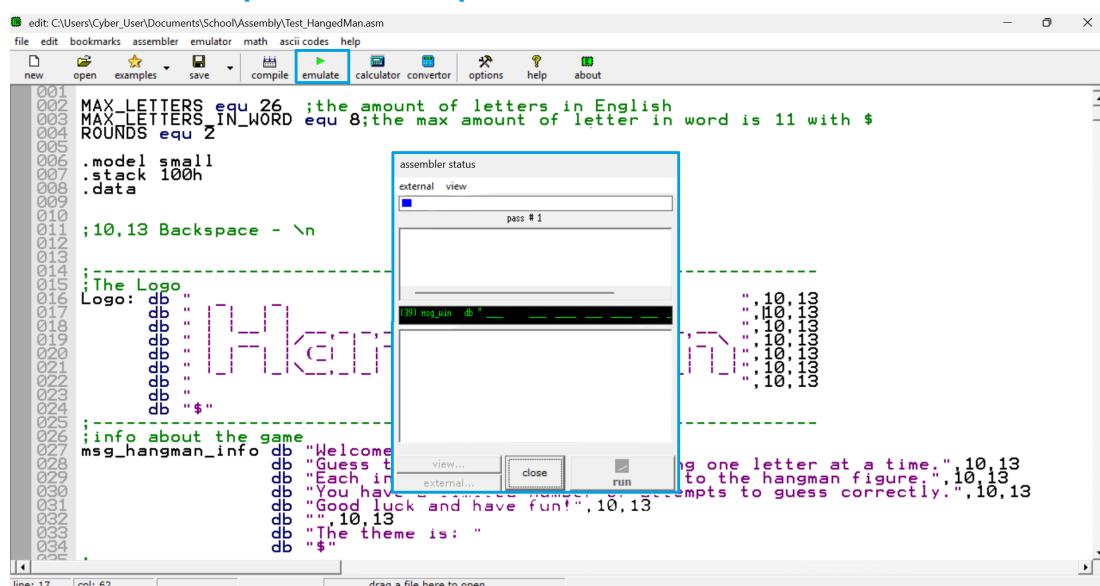
סביבת העבודה:

שפת התוכנה: Assembly סביבה העבודה: emu8086

סביבה 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.

צלומים של סביבת העבודה:

בשביל להריץ את הקוד צריך לחוץ על **emulate** כשלוחצים על **הקמפל של הקוד**



אם הקוד עבר את שלב הקומpileציה בהצלחה נפתחות שני חלונות
האלה:

The screenshot shows a debugger interface with two main windows. The left window is titled 'emulator: Test_HangedMan.exe_...' and displays assembly code, registers, and memory dump. The right window is titled 'original source code' and shows the C-like pseudocode used to generate the assembly. The assembly code pane highlights the instruction 'MOV AX, @data'.

```

;msg_get_char_input
msg_letter_already_selected db ?
msg_not_in_range db ?
msg_hidden_guess db ?
msg_incorrect_guess db ?
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 again? $"

;Variables
randomNum dw 0
old_letters db MAX LETTERS dup ?
selectedWord dw ?
hiddenWord db MAX LETTERS_IN_WORD
num_hidden_letter db 0

.code
mov ax, @data

```

חלון כחול:

- להריץ את כל הקוד
- להריץ שורה אחת
- להחזיר שורה אחרת
- להריץ את הקוד מחדש

חלון ירוק:

- מופיע שורות הקוד של התוכנית
- הפס הצהוב הוא הקוד שמצביע על הפעולה הבאה לביצוע

- רשימת האוגרים
- הקוד בשפת מכונה
- הקוד באסמבלי
- רשימת הדגלים
- המחסנית
- רשימת משתנים
- מיקום הקוד בזיכרון

The screenshot shows the HxD debugger interface. The assembly window displays assembly code with highlighted instructions. The registers window shows CPU register values. The stack window shows the stack memory. The flags window shows the state of various processor flags. The variables window shows global variables defined in the code.

מקטעים:

גודל כל מקטע הוא 64k.
 מקטע הקוד - Code segment
 מקטע הנתונים - Data segment
 מקטע המחסנית - Stack segment
 מקטע הרזרבי (אם אין מספיק מקום מוקם במקטעים האחרים משתמשים בו) - Extended segment.

הדגלים:

- FR - אוגר הדגלים, בן 16 סיביות המכיל 10 דגלים.
השתמשתי ב 3 דגלים:
S - דגל הסימן - נדלק שתוצאת פעולה אритמטית היא שלילית.
Z - דגל האפס - נדלק שתוצאת פעולה אריתמטית היא 0.
C - נדלק שיש שארית או שפעולה אריתמטית כורגת מתחום הזיכרון.

האוגרים:

האוגרים יכולים להיות אחדות ואפסים ארבעת האוגר המרכזים הם 16 סיביות. שמתפרקם לשני אוגרים:

auge L (נמוך) 7-0 סיביות בזיכרון.

auge H (גובה) 15-8 סיביות בזיכרון.

$$AX = AH, AL$$

$$BX = BH, HL$$

$$CX = CH, CL$$

$$DX = DH, DL$$

IO - 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 מרأس 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 8086 נ书写 בכתובת **C0460** במקטע BIOS Data Segment נשמר שעון המערכת, שמו "טייקים" מאז חצotta. ניתן לגשת לערך זהה כדי לקבל מידע על הזמן או ליצור מספרים אקראיים. או בעזרת הפקה:

בקשה לקבל את הזמן הנוכחי ;
int 21h ;
 לקריאה לפקה ;
 ; מאיות השנייה = DL, דקות = DH, שעה = CL

הגדרת המשתנים:

קבועים(equ):

MAX LETTERS מציין את מספר האותיות באנגלית -

MAX LETTERS IN WORD מציין את כמות האותיות המksamלית שיש במילה -

ROUNDS מציין את כמות הסיבובים -

משתנים המכילים הודעות(מחזרות) להדפסה:

Logo מכיל את הלוגו בתוי אסק' -

msg_hangman_info מכיל את הסבר המשחק -

msg_win מכיל את הودעת הפסד בתוי אסק' -

msg_lose מכיל את הודעת הפסד בתוי אסק' -

msg_Good_Bye מכיל גוד בי' בתוי אסק' -

msg_get_char_input מכיל הכנסתו -

msg_letter_already_selected מכיל האות כבר נבחרה -

msg_not_in_range מכיל אות לא טווח -

msg_hidden_guess מכיל ניחשת את נוכנה -

msg_incorrect_guess מכיל ניחוש לא נכון בהצלחה בפעם הבאה -

msg_round מכיל ראונד -

msg_wrong_guesses מכיל ניחושים לא נכוןים -

msg_show_word מכיל המילה היא -

msg_another_game אתה רוצה עוד משחק -

משתנים שם לא להדפסה:

fruits0 - fruits19 מכילים מילים מהנושא פירות -

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 תלוי -

words מערך של מצביעים שמכיל את כל המילים האפשרות -

topics - מערך של מצביעים לכל הנושאים האפשרים -

Hanged_man_index משתנה 8 סיביות, מכיל כמה פעמים הודפס האיש תלוי -

randomNum משתנה 16 סיביות, מכיל את המספר הרנדומלי -

old_letters מערך 8 סיביות, מכיל את כל האותיות שניחסו -

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 מושימות של הפרוייקט:

להדפיס לוגו למסך וסביר על המשחק.

להציג מילה ונושא רנדומלי.

ולהדפיס נושא רנדומלי.

להדפיס את כמות הסיבובים.

להדפיס את האותיות הלא נכונות.

לעדרן מילה מוסתרת עם "_" ועם האותיות שניחשו.

להדפיס מילה מוסתרת.

בודק ניצחון(כל האותיות במילה המוסתרת גלוויות).

לקלוטתו מהמשתמש.

לבזוק שהתו הוא אות קטנה באנגלית.

לבזוק שהאות המנוחשת לא נמצא כבר במילים המוסתרות.

לבזוק שהאות המנוחשת נמצאת במילה .

להדפיס את האיש התליי 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 ומידע על המשחק, שם ב <code>random</code> מספר רנדומלי בין 0-59-0 מגיריל ומדפיס נושא שם את המילה המוגרלת <code>selectedWord</code>	מדפיס לוגו למסך, מספיס מידע על המשחק, מדפיס נושא מוגרל
print_string	מקבל מצביע למחוזת למסך מהמחסנית	מדפיס את המילה מהמחוזת למסך	מדפיס את המילה מהמחסנית למסך, מוסיף 2 למחסנית כדי לנוקוט את המקום שהתקבל
updateHiddenWord	כלום	מלא את המילה המוסתרת ב "_" אם עדין לא ניחשו את האות במילה ובאותו שניחשו אותה	מלא את המילה המוסתרת ב "_" אם עדין לא ניחשו את האות במילה ובאותו שניחשו אותה
check_win	מקום שמור במחסנית	מחזיר 1 אם השחקן ניצח 0 אם לא דרך המחסנית	בודק שככל התווים ב <code>hidden_word</code> אינם " <code>_</code> "
check_correct_guess	מקום שמור במחסנית	מחזיר 1 אם המשתמש ניחש אות נכון אחרת 0 מחזיר 0 דרך המחסנית	בודק אם כמות התווים שהם לא " <code>_</code> " גדול מ <code>num_hidden_letter</code>
get_input	מקום שמור במחסנית	מחזיר 1 אם האות בין Z-a וגם בודק אם האות לא נוחשה כבר אם היא לא נוחשה כבר אחרת 0 דרך המחסנית	בודק אם האות היא בין Z-a וגם בודק אם האות לא נוחשה כבר אם היא לא נוחשה כבר אחרת בתנאים מדפיס הודעה שגיאה
get_char	מקום שמור במחסנית	מחזיר את התו שהמשתמש הכניס דרך המחסנית	המשתנן מכניסתו בעזרת פסקה, יורד שורה ומছיר את התו דרך המחסנית

בודק אם התו שהוכנו בטוויח אסקי בין 97 - 122 אם כן מוחזיר 1, אחרת מוחזיר 0 דרך המחסנית, מוסף 2 למחסנית כדי לנוקות את המקום שהתקבל	מחזיר 1 אם התו בטוויח בין z-a אחרית מוחזיר 0 דרך המחסנית	מקבלתו דרך המחסנית, שומר מקום במחסנית	check_illegal_letter
בודק אם האות לא נמצאת במערך של האותיות שנחקרו ומוחזיר אחד אם האות לא נמצאת, אחרת מוחזיר 0 דרך המחסנית, מוסף 2 למחסנית כדי לנוקות את המקום שהתקבל	מחזיר 1 אם האות לא נוחשא כבר אחרת מוחזיר 0	מקבלתו דרך המחסנית, שומר מקום במחסנית	check_letter_exists
מוסיף אות למערך של הניחושים(אחרי האות الأخيرة שנוחשאה), מוסף 2 למחסנית כדי לנוקות את המקום שהתקבל	מוסיף אות למערך של הניחושים	מקבלתו דרך המחסנית	add_letter
עובר על כל המערך של הניחושים ובודק שהאות לא נמצאת במילה שהוגירה	מדפיס את כל האותיות הלא נכונות שנחשו עם רווח בין כל אות	כלום	print_letter_guses
מקבל את הזמן מהשעון באמצעות פסיקה ומחלק אותו בערך מהחסנית ומוחזיר אותו דרך המחסנית, מוסף 2 למחסנית כדי לנוקות את המקום שהתקבל	מחזיר את מספר רנדומלי בין 0 - 1 לערך שהוכנו לפונקציה	מקבלתו דרך המחסנית, שומר מקום במחסנית	generate_random_number
מדפיס את התו 10 ואת התו 13 באמצעות פסיקה	מדפיס שורה חדשה	כלום	newline

אלגוריתם:

קרא לחת תוכנית menu
קרא לחת תוכנית updateHiddenWord
השם ב cx את_cmota קבוע ROUNDs

:main_game

שם b ו s את Hanged_man_index
מכפיל את s ב 2

:correct_letter_loop

מחסיר 2 מ sp
מכניס את kp לראש המחסנית

קורא ל check_win

השם b kp את sp

השם b kp את [kp+2] מהחסנית

השם b kp את הערך בראש המחסנית
מוסיף ל kp את הערך 2

אם 1 == kp

קובץ ל player_win

newline

קורא ל newline

השם b kp את מצביע ל msg_round

מכניס את kp לראש המחסנית

קורא ל print_string

השם b kp את kp

הוסף ל kp את '0'

השם b ah את הערך 2

קורא לפונקציה במיקום ah21

קורא ל newline

השם ב xp את מצביע ל msg_wrong_guesses
מכניס את xp לראש המחסנית
קורא ל print_string
קורא ל print_letter_guses
קורא ל newline
קורא ל newline

השם ב xp את מצביע ל hidden_word
מכניס את xp 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 xp את [2+bp] מהחסנית
השם ב bp את הערך בראש המחסנית
מוסיף ל ds את הערך 2
אם 1 != dp:
 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 xp את [2+bp] מהחסנית
השם ב bp את הערך בראש המחסנית
מוסיף ל ds את הערך 2

אם $1 == \text{dl}$
קופץ ל `correct_letter_loop`

השם ב `dp` את מצביע ל `msg_incorrect_guess`
מכניס את `dp` בראש המחסנית
קורא ל `print_string`
קורא ל `newline`

השם ב `dp` את מצביע ל `[Hanged_mans + si]`
מכניס את `dp` בראש המחסנית
קורא ל `print_string`
קורא ל `newline`

מוסיף אחד ל `x`
אם $0 != \text{cl}$
קופץ ל `main_game`
קורא ל `player_lose`

:`player_win`
קורא ל `newline`
קורא ל `newline`
השם ב `dp` את מצביע ל `msg_show_word`
מכניס את `dp` בראש המחסנית
קורא ל `print_string`
השם ב `dp` את מצביע ל `selectedWord`
מכניס את `dp` בראש המחסנית
קורא ל `print_string`
קורא ל `newline`
קורא ל `newline`
השם ב `dp` את מצביע ל `msg_win`
מכניס את `dp` בראש המחסנית
קורא ל `print_string`
קורא ל `newline`

קופץ ל skip_lose

:player_lose

קורא ל newline

קורא ל newline

השם ב xp את מצביע ל msg_show_word

מכניס את xp לראש המחסנית

קורא ל print_string

השם ב xp את מצביע ל selectedWord

מכניס את xp לראש המחסנית

קורא ל print_string

קורא ל newline

קורא ל newline

השם ב xp את מצביע ל msg_lose

מכניס את xp לראש המחסנית

קורא ל print_string

קורא ל newline

:skip_lose

השם ב xp את מצביע ל msg_Good_Bye

מכניס את xp לראש המחסנית

קורא ל print_string

סוף

תחילת תת תוכנית 1 (menu)

מכניס את כל האוגרים לראש המחסנית

השם ב xp את מצביע ל Logo

מכניס את xp לראש המחסנית

קורא ל print_string

קורא ל newline

השם ב ax את מצביע ל msg_hangman_info
מכניס את ax לראש המחסנית
קורא ל print_string

מחסיר 2 מ ds
מכניס את kp לראש המחסנית
מכניס את הערך 20 לראש המחסנית
קורא ל generate_random_number
השם ב kp את dp
השם ב ax את [2+kp] במחסנית
השם ב kp את הערך בראש המחסנית
מוסיף ל dp את הערך 2
השם ב randomNum את ax

מחסיר 2 מ dp
מכניס את kp לראש המחסנית
מכניס את הערך 3 לראש המחסנית
קורא ל generate_random_number
השם ב kp את dp
השם ב ax את [2+kp] במחסנית
השם ב kp את הערך בראש המחסנית
מוסיף ל dp את הערך 2

השם ב os את ax
מכפיל ב 2 את os
השם ב ax את הכתובת של [os+topics] ממוקטע הזיכרון
מכניס את ax לראש המחסנית
קורא ל print_string
קורא ל newline

השם ב ax ערך 20
הכפל ax ב ax

הוסף ל `randomNum` את `ax`
השם ב `si` את `randomNum`
הכפל אס ב 2
השם ב `ax` מבצע `si + words` ממקטע הזיכרון
השם ב `selectedWord` את `ax`

מציא כל האוגרים מראש המחסנית
קופץ לכתובת שבראש המחסנית
סוף תת תוכנית 1 (menu)

תחילה תת תוכנית 2 (print_string)
מכניס כל האוגרים לראש המחסנית
השם ב `dp` את `ds`
השם ב `ax` ערך `[18+dp]` מהמחסנית

השם ב `ah` את הערך 9
קורא לפונקציה במיקום `h21`

מציא כל האורים מראש המחסנית
הוסף 2 ל `ds`
קופץ לכתובת שבראש המחסנית
סוף תת תוכנית 2 (print_string)

תחילה תת תוכנית 3 (updateHiddenWord)
מכניס כל האוגרים לראש המחסנית
השם ב `si` את המצביע `selectedWord`
השם ב `ax` את המצביע `old_letters`
:loop_process_word
השם ב `ai` את הערך שצביע עליו `si`
ב `di` את הערך שצביע עליו `old_letters`
:check_guessed_letters

אם הערך שבכותרת `di` שווה ל 0:
 `կופץ ל add_underscore`
 השם ב `ah` את הערך שמצויבע עליו `di`
 אם `ah == al`
 `կופץ ל add_char_letter`
 מוסיף ל `di` אחד
 `կופץ ל check_guessed_letters`
 `:add_char_letter`

 השם ב `ah` את הערך שמצויבע עליו `ax`
 `կופץ ל continue_processing`
 `:add_underscore`
 השם בערך שמצויבע עליו `ax` את '_
 `:continue_processing`

 מוסיף ל `os` אחד
 מוסיף ל `ax` אחד
 אם הערך שבכותרת `os` לא שווה ל '\$':
 `կופץ ל loop_process_word`
 השם בערך שמצויבע `ax` את '\$'

מוציא את כל האוגרים מראש המחסנית
 `կופץ לכותרת שבראש המחסנית`
סוף תת תוכנית 3 (updateHiddenWord)

תחילת תת תוכנית 4 (check_win)

מכניס את `os` לראש המחסנית
מכניס את `ax` לראש המחסנית

 השם ב `os` את המצביע של `hidden_word`
 השם ב `di` את הערך 1
 `:check_win_loop`

השם ב או את הערך שמצויבע עליוosi
אם או לא שווה ל '_':
skip_check_win_loop
השם ב dl את הערך 0
:skip_check_win_loop
הוסף לוosi אחד
אם או לא שווה ל '\$':
kopfz ל check_win_loop
השם ב [8+k] במחסנית את הערך ax

מציא את הערך שבראש המחסנית ל ax
מציא את הערך שבראש המחסנית לוosi
kopfz ל כתובת שבראש המחסנית
סוף תת תוכנית 4 (check_win)

(check_correct_guess)

מכניס את כל האוגרים לראש המחסנית
hidden_word
השם ב לוosi את המצויבע ל hidden_word
השם ב dl את הערך -1
השם ב dl את הערך 0
:proces_word
השם ב או את הערך שמצויבע עליוosi
אם או שווה ל '_':
kopfz ל add_num_hidden_letter
מוסיף לוosi אחד
:add_num_hidden_letter
הוסף לוosi אחד
אם או לא שווה ל '\$':
proces_word
משווה את dl ל num_hidden_letter
אם dl קטן או שווה מ num_hidden_letter
skip_hidden_letter

השם ב `hidden_letter` מוח את הערך `a`
השם ב `a` את הערך 1
השם ב `cx` את המצביע על `msg_hidden_guess`
מכניס את `cx` בראש המחסנית
קורא ל `msg_hidden_guess`
newline
:skip_hidden_letter
השם ב `dp` את `sp`
השם ב `[20 + dp]` במחסנית את `xp`

מציא את כל האוגרים מראש המחסנית
קופץ לכטובת שבראש המחסנית
סוף תת תוכנית 5 (check_correct_guess)

תחילת תת תוכנית 6 (get_input)
מכניס את כל האוגרים לראש המחסנית

השם ב `xp` את מצביע על `input`
מכניס את `xp` בראש המחסנית
קורא ל `print_string`

מחסיר 2 מ `sp`
מכניס את `dp` לראש המחסנית
קורא ל `get_char`
השם ב `dp` את `sp`
השם ב `xa` את `[2+dp]` במחסנית
השם ב `dp` את הערך בראש המחסנית
מוסיף ל `ds` את הערך 2

מכניס את `xa` לראש המחסנית
מחסיר 2 מ `sp`
מכניס את `dp` לראש המחסנית

מכניס את ax לראש המחסנית
קורא ל check_illegal_letter
השם ב bp את ds
השם ב ax את [2+bp] במחסנית
השם ב bp את הערך בראש המחסנית
מוסיף ל ds את הערך 2
מושך את ax מראש המחסנית

אם $1 \neq !\text{bl}$:
skip_check_already_selected

מחסיר 2 מ ds
מכניס bp לראש המחסנית
מכניס ax לראש המחסנית
קורא ל check_illegal_letter
השם ב bp את ds
השם ב ax את [2+bp] במחסנית
השם ב bp את הערך בראש המחסנית
מוסיף ל ds את הערך 2

אם $0 \neq !\text{bl}$:
skip_print
השם ב xp את מצביע ל msg_letter_already_selected
מכניס xp לראש המחסנית
קורא ל print_string
קפוץ ל skip_print

:skip_check_already_selected
השם ב xp את מצביע ל msg_not_in_range
מכניס xp לראש המחסנית
קורא ל print_string
:skip_print
השם ב bp את ds

השם ב $[20 + \text{dp}]$ במחסנית את a
מציא את כל האוגרים מראש המחסנית
קופץ לכתובת שבראש המחסנית
סוף תת תוכנית 6 (get_input)

(get_char)

מכניס את ax לראש המחסנית
מכניס את ax לראש המחסנית
מכניס את ds לראש המחסנית

השם ב ah את הערך 1
קורא לפונקציה במיקום $\text{h}1$
קורא `newline`

השם ב dp את ds
השם ב $[10 + \text{dp}]$ במחסנית את הערך ax

מציא את הערך שבראש המחסנית ל sp
מציא את הערך שבראש המחסנית ל bx
מציא את הערך שבראש המחסנית ל ax
קופץ לכתובת שבראש המחסנית
סוף תת תוכנית 7 (get_char)

(check_illegal_letter)

מכניס את כל האוגרים לראש המחסנית
השם ב dp את ds
השם ב ai ערך $[18 + \text{dp}]$ מהמחסנית
השם ב ax את הערך 0

אם $97 < \text{ai} :$
קופץ ל `latter_not_in_range`

אם $122 > al$

קופץ ל `latter_not_in_range`

השם ב `ls` את הערך 1

:`latter_not_in_range`

השם ב $[22+bp]$ במחסנית את `cx`

מוציא את כל האוגרים מראש המחסנית

הוסף 2 ל `ds`

קופץ לכטובת שבראש המחסנית

סוף תת תוכנית 8 (check_illegal_letter)

תחילת תת תוכנית 9 (check_letter_exists)

מכניס את כל האוגרים מראש המחסנית

השם ב `dp` את `ds`

השם ב `xp` ערך $[18+dp]$ מהמחסנית

השם ב `os` את המצביע על `old_letters`

:`check_loop`

אם הערך שבכטובה זו שווה ל 0:

קופץ ל `letter_not_found`

אם או שווה לערך שבכטובה זו:

קופץ ל `letter_found`

מוסיף ל `os` אחד

קופץ ל `check_loop`

:`letter_found`

השם ב `bx` את הערך 0

קופץ ל `skip2`

:`letter_not_found`

מכניס את `ax` לראש המחסנית

קורא ל `add_letter`
השם ב `ax` את הערך 1
`:skip2`

השם ב `[bp+22]` במחסנית את `ax`
מציא את כל האוגרים מראש המחסנית
הוסף 2 ל `ds`

קופץ לכטובת שבראש המחסנית
סוף תת תוכנית 9 (check_letter_exists)

תחילת תת תוכנית 10 (add_letter)
מכניס את כל האוגרים לראש המחסנית
השם ב `dp` את `ds`
השם ב `ax` ערך `[bp+18]` מהמחסנית
השם ב `cx` את הערך 0

השם ב `os` את מצביע ל `old_letters`
:check_add_letter_loop

אם הערך שבכטובת `os` שווה ל 0:
קופץ ל `skip_check_add_letter_loop`
הוסף ל `os` אחד
הוסף ל `cx` אחד
קופץ ל `check_add_letter_loop`
:skip_check_add_letter_loop

אם `cx` שווה לקבוע `:MAX LETTERS`
קופץ ל `letter_arr_full`
השם `ax` את המצביע ל `old_letters`
הוסף `ax` ל `cx`
השם בערך `ax` מצביע עליו את `al`

:letter_arr_full

מציא את כל האוגרים מראש המחסנית
הוסף 2 ל **ds**

קופץ לכתובת שבראש המחסנית
סוף תת תוכנית 10 (add_letter)

(print_letter_guses)

מכניס את כל האוגרים לראש המחסנית
השם ב **ax** את המצביע ל **old_letters**
:loop_old_letters

השם ב **ch** את הערך ש **ax** מצביע עליו
השם ב **cl** את הערך **1**
השם ב **os** את המצביע של **selectedWord**
:loop_selectedWord:

השם ב **ao** את הערך ש **os** מצביע עליו
מוסיף ל **os** אחד
אם **ch** לא שווה ל **al**:
קופץ ל **cl** change
השם ב **cl** את הערך **0**
:skip_change_cl

אם **ao** לא שווה ל **'\$'**
קופץ ל **loop_selectedWord**
אם **1 != cl**:
קופץ ל **print_char**
השם ב **cl** את **ch**
השם ב **ah** את **2**
קורא לפונקציה במיקום **21h**
השם ב **ap** את הערך **'_'**

קורא לפונקציה במיקום 21h
:print_char

הוסף ל ax אחד
אם !=0 ch
קופץ ל loop_old_letters

מציא את כל האוגרים מראש המחסנית
קופץ לכתובת שבראש המחסנית
סוף תת תוכנית 11 (print_letter_guses)

תחילה 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 kp את ds
השם ב ax את xp
השם ב xp את הערך 0
השם ב xp ערך [bp+18] מהמחסנית
לחلك ax ב xp

השם ב kp את ds
השם ב [bp+22] מהמחסנית את הערך xp

מציא 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 ap לראש המחסנית

השם ב ah את הערך 02h

השם ב af את הערך 10

庫ורא לפונקציה במיקום ah21

השם ב af את הערך 13

庫ורא לפונקציה במיקום ah21

מציא את ap לראש המחסנית

מציא את ax לראש המחסנית

קופה לכתובת שבראש המחסנית

סוף תת תוכנית 13 (newline)

סוף פרויקט

```
;-----  
; 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 " \V V/ || |VV| |VV| | | / ", 10,13

db " \wedge / \sqcup \sqcap \sqsubseteq \sqsupseteq \sqcap \sqcap \sqsubseteq \sqsupseteq / \sqcup \sqcap \sqsubseteq ",
10,13

```
db "      \\",  
_____|_____|\\_____|_____|\\_____|_____|_____|_____|_____|_____|",  
10.13
```

```
    db "YOU WIN :)", 10,13  
    db "", 10,13
```

msg_lose:

```
db "
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 "\$"

j

; 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
```

Hanged_man2:

```
db " _____ ", 10,13  
db " |    |", 10,13  
db " |    O", 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           ;variable for random number
old_letters db MAX LETTERS dup(0) ;26 cells with value 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 correct 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 correct_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 beginning 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 adres 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 correct else 0.
```

```
; Input: From the stack.
```

```
; Output: Return to the stack 1 if the legal and correct, 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 return 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 letter 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 because 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;amount 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 between (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 with 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
```

תמונה מהתוכנית:

הפט

[H] [A] [N] [G] [M] [A] [N] [G]
[I] [E]

Welcome to Hangman!
Guess the hidden word by selecting one letter at a time.
Each incorrect guess adds a part to the hangman figure.
You have a limited number of attempts to guess correctly.
Good luck and have fun!

The theme is: Cities!

Round: 7
Wrong Guesses:

Enter a letter:

```
Round: 5
Wrong Guesses: a z

____n
Enter a letter: q
Wrong guess! Better luck next time!
-----
|   |
|   O
|   |
-----
Round: 4
Wrong Guesses: a z q

____n
Enter a letter:
Letter not in range a-z, Enter a letter:
```

```
Round: 4
Wrong Guesses: a z q

____n
Enter a letter:
Letter not in range a-z, Enter a letter: h
Wrong guess! Better luck next time!
-----
|   |
|   O
|   |
-----
Round: 3
Wrong Guesses: a z q h

____n
Enter a letter:
```

```
Round: 3
Wrong Guesses: a z q h

____n
Enter a letter: b
You guessed a correct letter!

Round: 3
Wrong Guesses: a z q h
__b_n
Enter a letter: o
You guessed a correct letter!

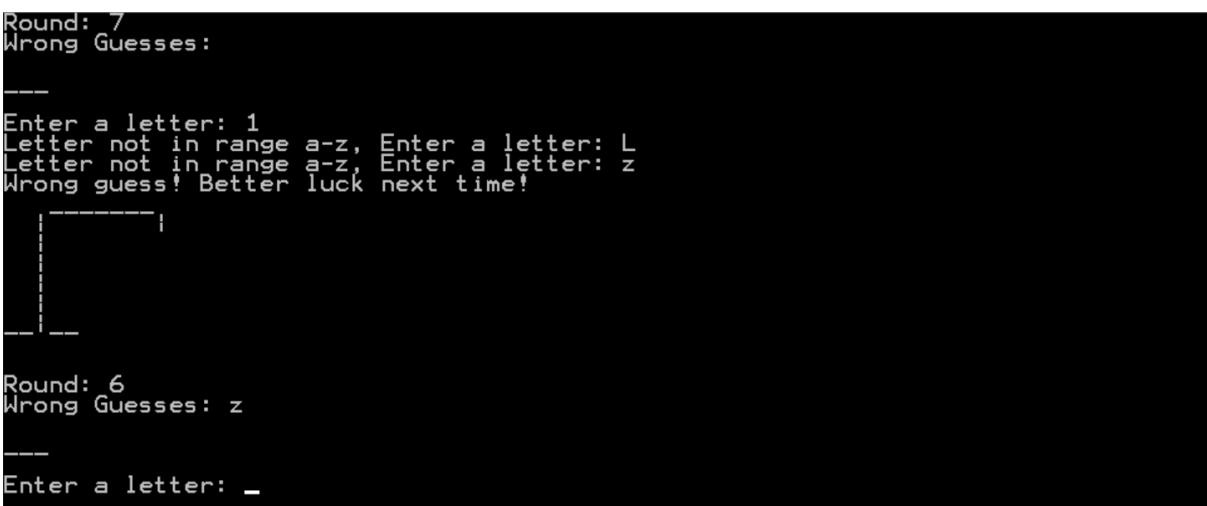
Round: 3
Wrong Guesses: a z q h
__bon
Enter a letter: _
```

Wrong guess! Better luck next time!





ניצחון:



```
Round: 6
Wrong Guesses: z

_____
Enter a letter: z
Letter already chosen, Enter a letter: .
Letter not in range a-z, Enter a letter: r
Wrong guess! Better luck next time!
```

```
Round: 5
Wrong Guesses: z r

_____
Enter a letter: _
```

```
Round: 5
Wrong Guesses: z r

_____
Enter a letter: f
You guessed a correct letter!

Round: 5
Wrong Guesses: z r
f__

Enter a letter: i
You guessed a correct letter!

Round: 5
Wrong Guesses: z r
fi_

Enter a letter:
```

```
Enter a letter: i
You guessed a correct letter!

Round: 5
Wrong Guesses: z r
fi_

Enter a letter: g
You guessed a correct letter!
```

```
The word is: fig

F I G U R E S O N D E R
Y O U   W I N  :)
```

The word is: fig



Y O U W I N :)



GOOD BYE! SEE YOU NEXT TIME