פעולות אריתמטיות לוגיות

# חיבור, חיסור, כפל וחילוק

בתרגיל זה ניצור מחשבון שיקבל כקלט שני מספרים חיובים גדולים מ-0, בני ספרה אחת (1 -9) ויציג את סכום המספרים, הפרשם, המכפלה שלהם והמנה שלהם.

מאחר ולא למדנו מספרים שלילים מחשב זה לא יציג מספרים שלילים.

**רשימת פסיקות - מחסן פקודות**

|  |  |
| --- | --- |
| פקודה להדפסת מחרוזת  (שם משתנה המחרוזת msg) | mov dx, offset msg  mov ah, 9h  int 21h |
| פקודה להדפסת תו על המסך | mov dl, 'x' ; print x  mov ah, 2  int 21h |
| פקודה למעבר שורה | mov dl, 0ah  mov ah, 2h  int 21h |
| פקודה לקליטת מספר מהמשתמש ושמירתו לרגיסטר al | mov ah, 1h  int 21h  sub al, '0' |
| פקודה להדפסת ערך השמור ברגיסטר **al** בתרגום הערך למספר בבסיס 10 על המסך  Ten הוא שם משתנה המכיל את הערך 10  ten db 10 | mov ah, 0  div [ten]  add ax, '00'  mov dx, ax  mov ah, 2h  int 21h  mov dl, dh  int 21h |

**הוראות כלליות לכתית התכנית:**

* לנוחותכם ולנוחות המורים יש לתעד כל פקודה או אוסף פקודות.
* עברו על ההוראות לפי השלבים. בסיום כל שלב הריצו ובדקו את התכנית. אם התכנית עבודת המשיכו הלאה. אם לא בדקו מה לא תקין, תקנו והמשיכו הלאה.
* בסיום כל שלב שמרו עותק חדש , והוסיפו את הקוד לקובץ החדש.   
  במקרה והתכנית לא תעבוד תמיד תוכלו לחזור שלב אחד אחורה ולהמשיך משם.

**השלבים ליצירת המחשבון.**

1. בתכנית שלנו שני משתנים עבור מספרי הקלט. צרו שני משתנים בגודל בית [num1], [num2]

ומשתנה נוסף בגודל בית - **משתנה** **עזר** לצורך הדפסת תוצאת החישוב - [ten]

ten db 10

1. צרו משתנה מחרוזות להוראות למשתמש. (איזה נתונים להקליד)

input\_text db 'Enter one digit number: $'

1. הפסיקה להדפסת מחרוזת היא:

; print message to screen

mov ah, 9h

int 21h

כתבו פעולה המדפיסה למסך את מחרוזת.

לפני הזימון העבירו לרגיסטר dx את מיקום תחילת המחרוזת

mov dx, offset input\_text

זמנו את הפעולה מהתכנית הראשית עבור הדפסת ה input\_text

call print\_message

הריצו את התכנית ובדקו שה text מודפס

1. קליטת נתונים מהמקלדת לרגיסטר al.  
   כתבו פעולה readUserInput המקבלת נתון מהמקלדת לרגיסטר al. השתמשו בפסיקה הבאה הקולטת מספר ל al

; get user first input number to al

mov ah, 1h

int 21h

sub al, '0'

זמנו את הפעולה מהתוכנית הראשית. שימו לב 🎔 לא להשתמש ב pusha

ו - popa בפרוצדורה הזאת שכן ערך הקלט נמצא ב al.

1. נעביר את המספר מ al למשתנה num1.

; move user input to [num1]

mov [num1], al

1. לצורך נראות טובה יותר של הפרטים על המסך בהמשך נעבור לשורה הבאה

במסך.

כתבו פעולה בשם newline העוברת שורה וזמנו אותה. השתמשו בפסיקה הבאה למעבר שורה

; new line

mov dl, 0ah

mov ah, 2h

int 21h

שמרו את התכנית והריצו. בדקו בעזרת ה turbo debugger האם המספר שהקשנו נשמר במשתנה [num1].

1. חזרו על הפעולה לקליטת המשתנה עבור המשתנה השני, הצגת הודעה למסך, קליטת הנתון לרגיסטר al, המרת הנתון לערך המבוקש ושמירת הנתון למשתנה [num2].   
   שמרו את התכנית, בדקו אם עובדת העתיקו את ההוראות לקובץ חדש והמשיכו לעבוד בקובץ החדש. אם לא בדקו מה לא תקין תקנו והמשיכו.

1. נתחיל בפעולת החיבור. צרו משתנה חדש, מסוג מחרוזת, השימו בו את המילה Addition. והציגו למסך את ההודעה לפני הדפסת תוצאת החיבור.

add\_text db 'Addition: $'

השתמשו בפעולה להדפסת מחרוזת.

1. בצעו את פעולת החיבור בין שני המשתנים. תוצאת החיבור נשמרת לרגיסטר al.

; add operation

mov al, [num1]

add al, [num2]

1. הפסיקה המציגה למסך את הנתון השמור ברגיסטר al :  
   כתבו פעולה printNumber המדפיסה למסך את המספר השמור ברגיסטר al.

; print the result stored in AL

mov ah, 0

div [ten]

הריצו את התכנית, הקלידו נתונים למסך ובדקו את התוצאה.   
בדקו את התכנית בעזרת ה – Turbo Debugger.  
שמרו את התכנית לקובץ חדש והמשיכו לעבוד עם הקובץ החדש.

add ax, '00'

mov dx, ax

mov ah, 2h

int 21h

mov dl, dh

int 21h

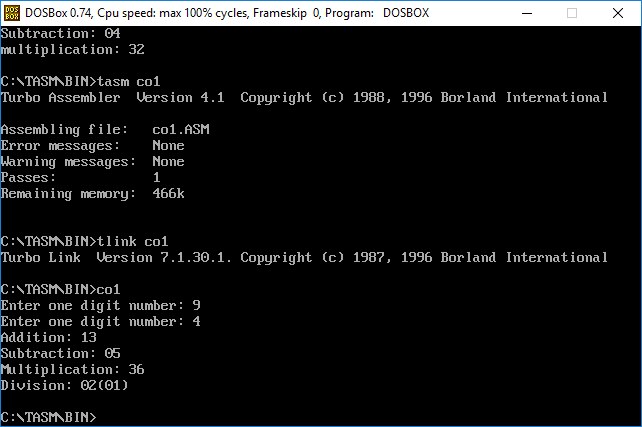
; call the procedure which prints new line

call newLine

1. נעבור לפעולת החיסור.  
   שמרו למשתנה הודעה מתאימה, הציגו אותה למסך, הציגו אותה למסך.  
   בצעו את פעולת החיסור ושמרו אותה לרגיסטר al.  
   הדפיסו למסך את הנתון השמור ברגיסטר al.  
   הריצו את התכנית, הקלידו נתונים למסך ובדקו את התוצאה. בדקו את התכנית בעזרת ה – Turbo Debugger. שמרו את התכנית לקובץ חדש והמשיכו לעבוד עם הקובץ החדש.
2. בצעו את הפעולות הדרושות לפעולת הכפל, והצגת התוצאה למסך.
3. בדקו אתה תכנית עם כמה מספרים. זכרו להקיש נתונים "נכונים" למחשבון שלנו. מספרים חד ספרתיים חיובים שלמים ושתוצאת החיסור לא תהיה שלילית.
4. בצעו את הפעולות הדרושות להצגת פעולת החילוק. זכרו פעולת החילוק נותנת את תוצאת החילוק בשלמים (תוצאת החילוק ב – 8 ביט נשמרת לרגיסטר al. הריצו ובדקו.  
   שימו לב 🎔, לפני פועלת החלוק יש לאפס את רגיסטר ax.
5. אם תרצו להציג את תוצאת החילוק + השארית עליכם להציג את תוצאת החילוק השמורה ברגיסטר al ואת השארית השמורה ברגיסטר al.

*כדי לא לאבד את תוצאת החילוק והשארית שמרו את רגיסטר al ו ah - למשתנים. לפני הזימון להדפסת התוצאה.*

היעזרו בפסיקות שבראש הדף להדפסה של תו אחד (כף תוכלו להציג את שארית החלוקה בתוך סוגרים. לדוגמה:



שמרו את התכנית והמשיכו בתכנית עם שם חדש.

1. נשכלל את המחשבון כך שנוסיף לו קריאת של הפעולה החשבונית שנרצה לבצע מהמקלדת

(\*, +, -) ונציג למסך רק את התוצאה של הפעולה החשבונית שהוקלדה.

ראשית ניצור מחרוזת הודעה למשתמש:

sign\_text db "Please enter operation sign: $"

ניצור משתנה חדש בו נשמור את התו שיסמן את הפעולה החשבונית

operation db ?

ניצור פעולה חדשה לקריאת הפעולה החשבונית מהמקלדת - **readOperation**

* נדפיס למסך את המחרוזת עם ההוראה להקלדת הסימן נתן הוראה למעבר שורה.
* נקלוט את התו מהמקלדת לרגיסטר al

; get user sing to AL

mov ah, 1h

int 21h

* נשמור את התו ברגיסר al **למשתנה** [operation]. אפשר בתוך הפעולה או בתוכנית הראשית לשיקולכם.   
  אל תשכחו להצהיר על המשתנה בתחילת התוכנית.
* נשלב את הזימון לפעולה זו בתוכנית הראשית במקום המתאים.   
  (לאחר קבלת שני מהספרים ולפני הדפסת התוצאה).

בתוכנית הראשית נבדוק מהו התו השמור במשתנה [operation] ועל פי התו השמור במשתנה נפנה לפעולה החשבונית המתאימה.

**לדוגמה בדיקה האם המשתנה הוא פעולת חיבור**

cmp [operation], '+'

הוראת השוואה וקפיצה לתווית אשר תפנה לפעולה של החיבור.

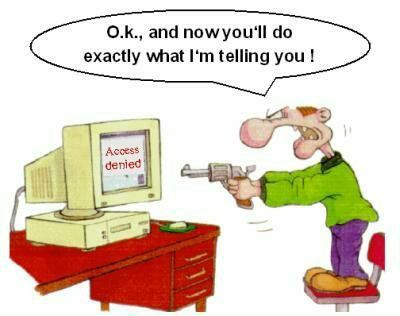
כשנחזור מהפעולה נרצה לקפוץ לתווית חדשה, המפנה להמשך התכנית (הדפסת התוצאה) לאחר בדיקות ההשוואה.

je Add\_op

Add\_op:

call addOp

**jmp endExecute**

בהצלחה.

קוד התכנית

IDEAL

MODEL small

STACK 100h

DATASEG

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

n1 db ?

n2 db ?

ten db 10

input\_text db 'Enter one digit number: $'

add\_text db 'Addition: $'

sub\_text db 'Subtraction: $'

mul\_text db 'Multiplication: $'

div\_text db 'Division: $'

rem\_n db ?

sign\_text db "Please enter operation sign $"

operation db ?

CODESEG

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

proc print\_message

; print input massege

mov ah, 9h

int 21h

ret

endp print\_message

proc get\_num

; get user input number to al

mov ah, 1h

int 21h

sub al, '0'

ret

endp get\_num

proc new\_line

; new line

mov dl, 0ah

mov ah, 2h

int 21h

ret

endp new\_line

proc print\_al

; print the result stored in AL

mov ah, 0

div [ten]

add ax, '00'

mov dx, ax

mov ah, 2h

int 21h

mov dl, dh

int 21h

ret

endp print\_al

start:

mov ax, @data

mov ds, ax

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

; print message and get 1st number

mov dx, offset input\_text

call print\_message

call get\_num

mov [n1], al

call new\_line

mov dx, offset input\_text

call print\_message

call get\_num

mov [n2], al

call new\_line

; print message and get 1st number

mov dx, offset sign\_text

call print\_message

call get\_num

mov [operation], al

call new\_line

; print message and get operation

mov dx, offset sing\_text ; the address of or message in register dx

call print\_message

; get user operation sign

mov ah, 1h

int 21h

mov [operation], al

call new\_line

; navigation to correct operation

cmp [operation], '+'

je Add\_op

cmp [operation], '-'

je Sub\_op

cmp [operation], '\*'

je Mul\_op

mp [operation], '/'

je Div\_op

jmp endExe

Add\_op:

; print Addition title

mov dx, offset add\_text

call print\_message

; add operation

mov al, [n1]

add al, [n2]

call print\_al

call new\_line

jmp endExe

Sub\_op:

print Subtraction title

mov dx, offset sub\_text

call print\_message

; sub operation

mov al, [n1]

sub al, [n2]

call print\_al

call new\_line

Mul\_op:

;print multiplication title

mov dx, offset mul\_text

call print\_message

; mul operation

mov al, [n1]

mul [n2]

call print\_al

call new\_line

Div\_op

; print Division title

mov dx, offset div\_text

call print\_message

; div operation

mov ax, 0

mov al, [n1]

div [n2]

חייבים להעביר את תוצאת שארית החלוקה ששמורה ברגיסטר ah למשתנה. בפסיקה של הדפסת מספר השמור ברגיסטר a יש שימוש ברגיסטר ah והמידע שהיה שמור בו נמחק.

**mov [rem\_n], ah**

call print\_al

;print remainder

mov dl, '(' ; print open (

mov ah, 2

int 21h

mov al, [rem\_n]

call print\_al

mov dl, ')' ; print closed )

mov ah, 2

int 21h

call new\_line

endExe:

exit:

mov ax, 4c00h

int 21h

END start