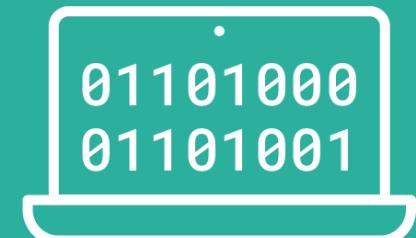




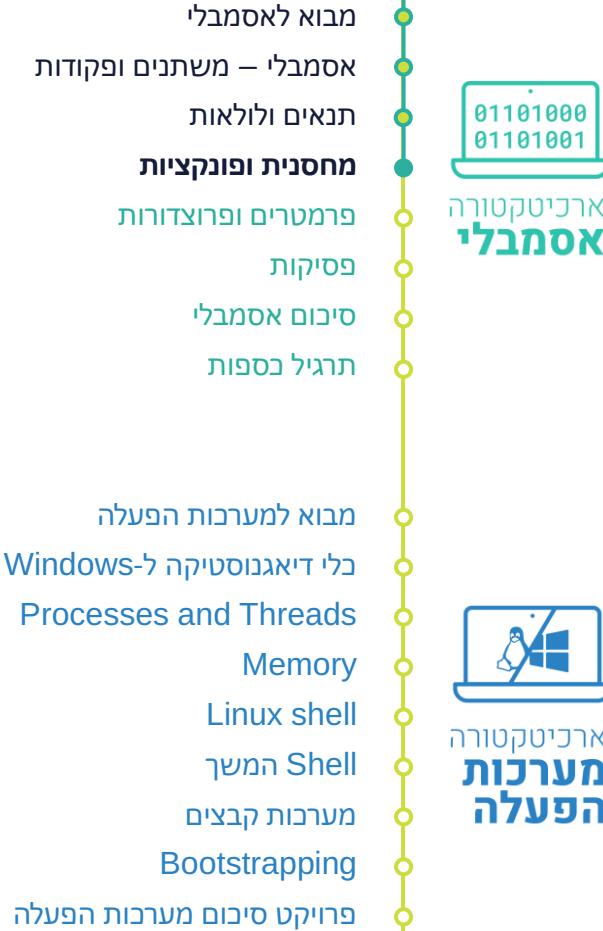
שיעור 4: המחשבונית ארQUITקטורה **אSEMBלי**



הಡיקו מצלמות
בבו מיקרופונים
השתיקו טלפונים
ארגנו מחברת וכל בຕיבה

שיעור 4

מחסנית ופונקציות





- למדנו שניתן לשלוט בתוכנית בעזרת תנאים.
- וגם כי כיצד עובדת לולה באסמבלי.
- תרגלנו את ההבנה שלנו בנסיבות השונות.
- כמו שאנחנו מכירים לולה היא לא הדרך היחידה לחסוך בקוד.

בשיעור הקודם

למדנו על תנאים ולולאות באסמבלי

שיעור 4

מחסנית ופונקציות

Error



Stack Overflow

סיכום

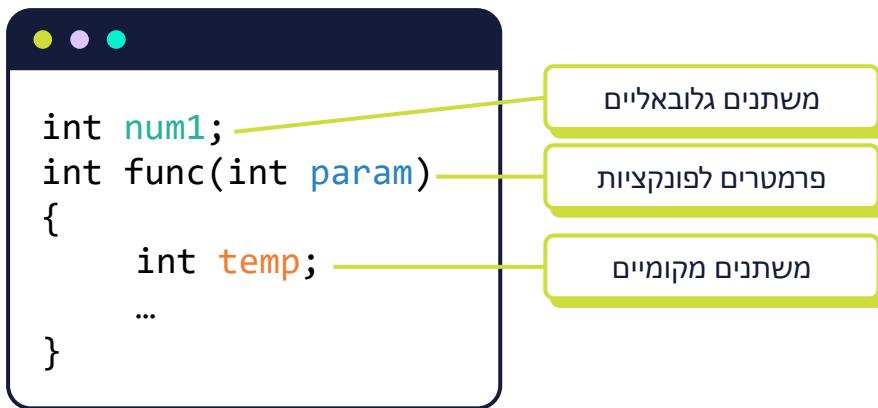
מסגרת

מבנה
פונקציה

פעולות
המחסנית

מבנה
המחסנית

בailo סוגים משתנים אנחנו משתמשים ב-C?



משתנים שהגדרכנו עד כה באסמלבי:
לאיזה שלושת הסוגים הנ"ל מתאימה הגדירה שבחז?

Num1 DW 3



מתי קיימן עותק יחיד ומתי יותר במשתנה מקומיים

זיכורת: למשתנה גלובלי עותק אחד ויחיד

משתנים מקומיים

יותר עותקים

אפו

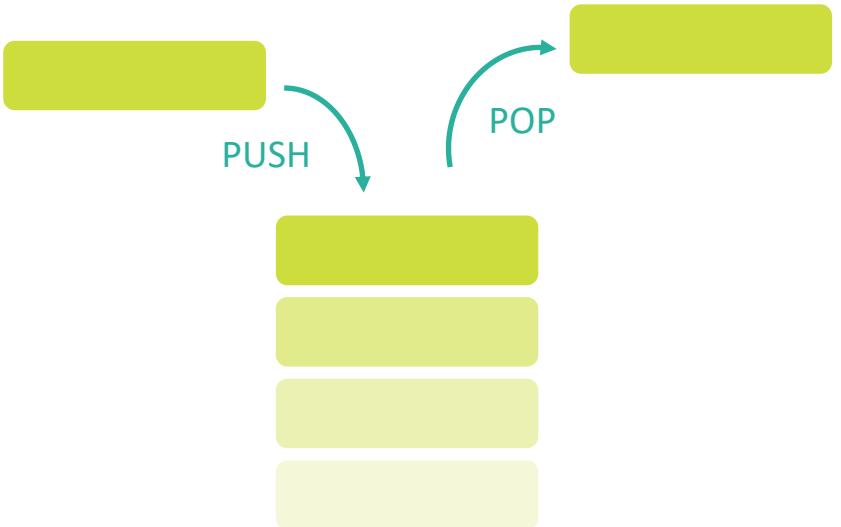
פונקציה רקורסיבית או
קוד שrex ב"מקביל"

אם הפונקציה לא נקראת
בכלל

- אנחנו צריכים להזכיר זיכרון **למשתנים מקומיים | פרמטרים** בזמן ריצת התוכנית.

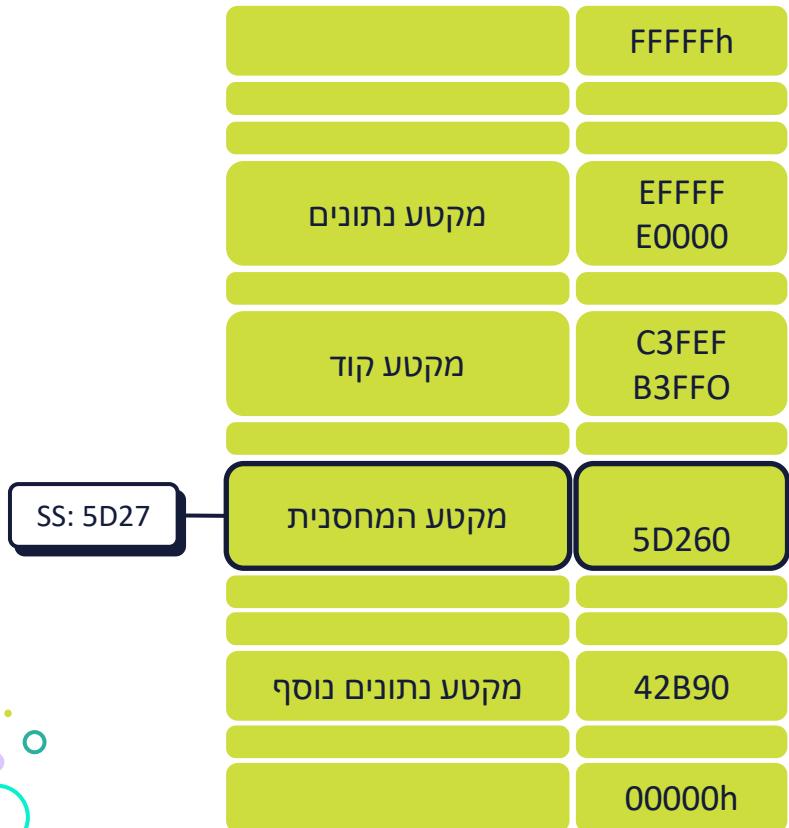
מחסנית Stack

- מבנה נתונים מאד נפוץ
- LIFO – Last In First Out
- שתי פעולות מרכזיות: PUSH, POP



מחסנית באSEMBLY

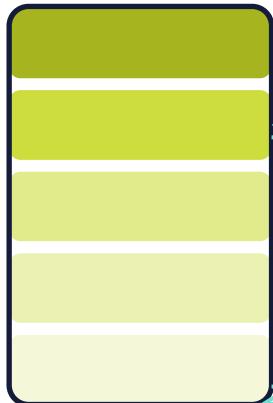
- משמשת לאחסון מידע לזמן קצר
- מקטע (segment) בזיכרון



מחסנית באסמבלי

לצורך עבודה נוחה עם זיכרון המחסנית קיימים שני אוגרים מיוחדים:

קטע המחסנית



היחס

אוגר SP (Stack Pointer)

מייצג את היחס (offset) מהתובות הבסיס

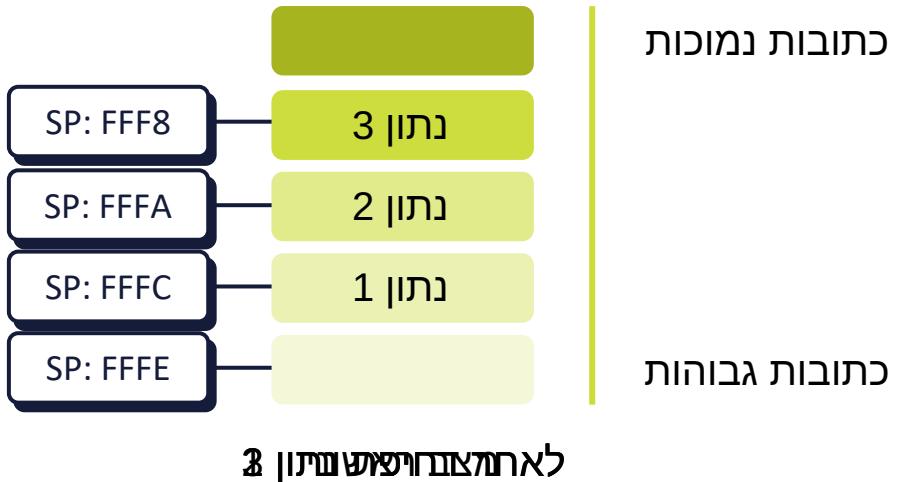
- בתחילת התוכנית, ערכו שווה לגודל המחסנית
- מתעדכן לאחר כל ביצוע של פקודות PUSH או POP

אוגר SS (Stack Segment)

מצביע על הכתובת הנמוכה ביותר
במחסנית

מחסנית באסמבלי

המצבייע לראש המחסנית (SP) קטן ולא גדול בכל שימושים אברים.





- לפקודות POP PUSH יש אופרנד אחד בלבד!
- גודלו של כל איבר במחסנית הוא קבוע, ולכן גודל האופרנד בפקודות הוא קבוע באסמבלי 8086, מדובר בשני בתים

האם הפקודה תקינה או לא?



שיעור 4

מחסנית ופונקציות



סיכום

מסגרת

מבנה
פונקציה

פעולות
המחסנית

מבנה
המחסנית

הפקודה

PUSH AX

לחצו לビיצוע הפקודה

PUSH AX

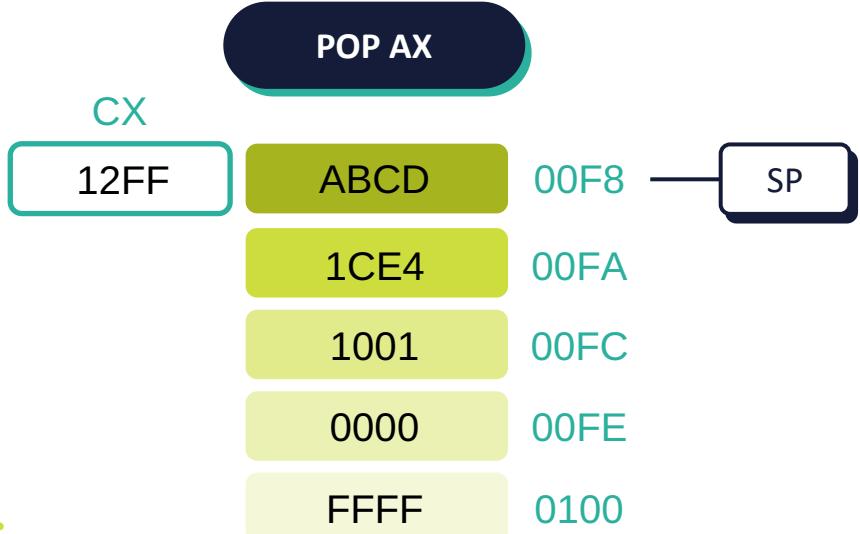
ABCD

| | |
|------|------|
| 00F8 | |
| 1CE4 | 00FA |
| 1001 | 00FC |
| 0000 | 00FE |
| FFFF | 0100 |

המצביע לראש המחסנית SP קטן ולא גדול
בכל שימושים אברים.

הפקודה POP AX

לחצו לビיצוע הפקודה





Stack Underflow

מה יעשה הקוד הבא?



לא ניתן לצפות. מסוכן!

על אחוריותו של המתוכנת לעבוד נכון עם המחסנית

מה דעתכם?
דברו

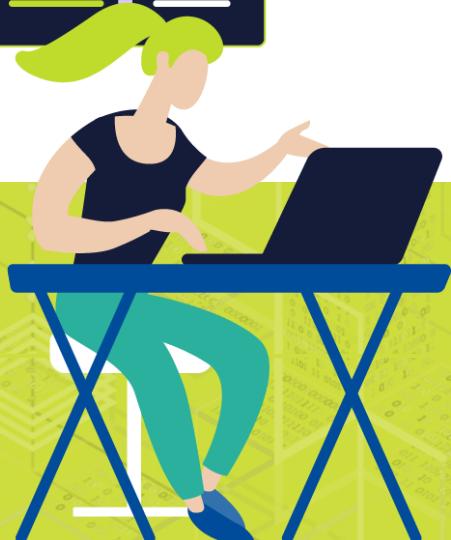
PUSHA / POPA

- .1 הורידו את הקובץ **addressing2.asm** מתיקית תרגיל ביתה.
- .2 התכנית שלפניכם משתמשת בפקודה **PUSHA**. קראו על פקודה זו (בספר / בקובץ הטקסט של הפקודות / באינטרנט ועוד').
- .3 הריצו את התוכנית step-by-step (שלב אחרי שלב), וודאו כי אתם מבינים את הפקודה.
- .4 השלימו את התוכנית על-פי ההוראות בר שטדים את המספרים הרצויים כאשר הערכים שלהם צריכים להילך מהמחסנית (אסור להשתמש במספרים מיידיים).
- .5 בסיום שחזורו את המחסנית ללא שימוש ב-**POPA** בר התוכנית תדפיס את המספר 23232 בהלהבה.
הקפידו לצייר את מצב המחסנית בכל שלב.



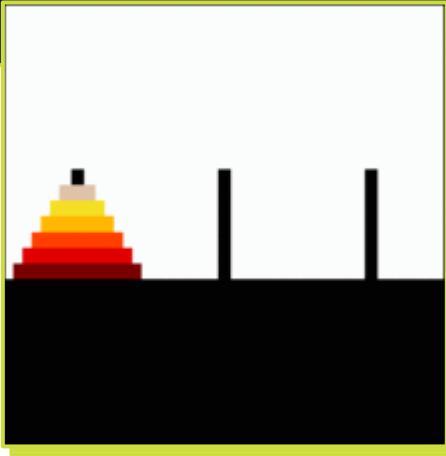
Live Coding

פיתוח בעורך קוד ↶



נדגים כל מיני דברים:

- מה עושה הפקודה PUSH
- מה סדר האוגרים שננדחפים
- נחזר את הערכים ללא שימוש בPOPA



Tower of Hanoi

מסופר כי במקדש בראהמי נמצאים בהנים אשר עוסקים בהעברת מגדל בין 64 דיסקיות. על פי האגדה, באשר הכהנים יסיוימו את עבודתם, הגיע סוף העולם.

air נחשב את בואו של סוף
העולם?

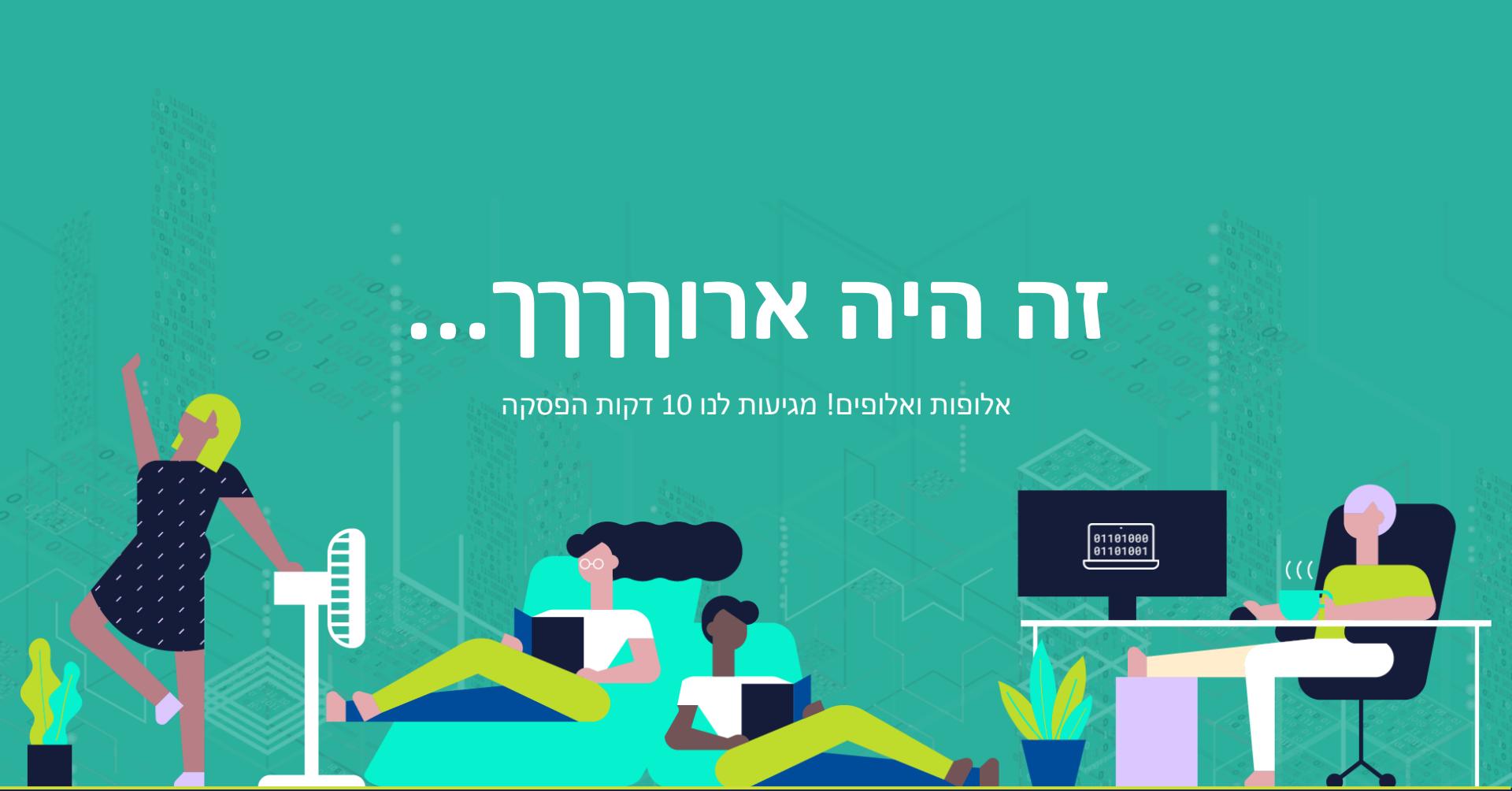
נוסחה כללית עבור מגדל בגובה n בלוקים: $2^n - 1$:
אז:

$$2^{64} - 1 = 18,446,744,073,709,551,615$$

מה דעתכם?
דברו

זה היה אורור...

אלופות ואלופים! מגיעות לנו 10 דקות הפסקה

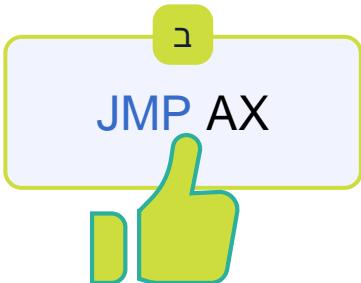




האם הפקודות תקינה או לא?

תזכורת: אוגר IP (Instruction Pointer)

- מחזק מצביע לכתובת של הפקודה הבאה אותה אנחנו רוצים להריץ
- IP היא כתובת היסט למקטע קוד (CS)
- לא ניתן לעדכן אותו ישירות, אלא רק ע"י המעבד



מ-C לאסמבלי

איך נממש (ונפעיל) פונקציה זו באסמבלי?

```
short sum(short x, short y)
{
    short result = x + y;
    return result;
}
```

מה דעתכם?
דברו

שיעור 4

מחסנית ופונקציות



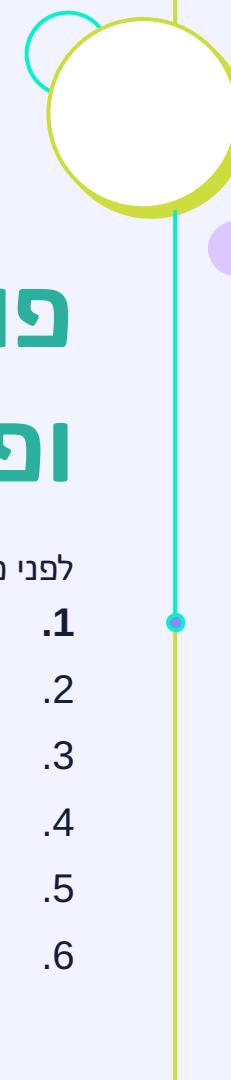
סיכום

מסגרת

מבנה
פונקציה

פעולות
המחסנית

מבנה
המחסנית



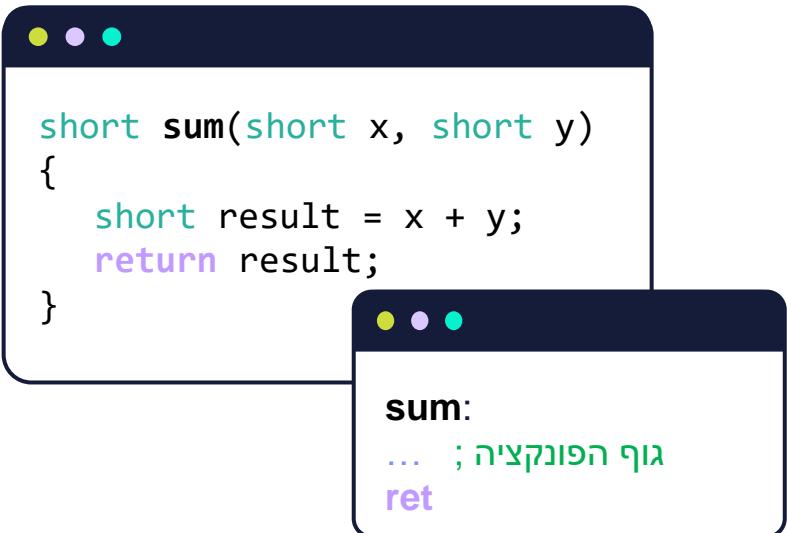
פונקציות ופרוצדורות

```
short sum(short x, short y)
{
    short result = x + y;
    return result;
}
```

לפני כן אנו צריכים לדעת תשובות למספר שאלות:

1. מהו המבנה של פונקציה?
 2. איך ניתן לקרוא (להפעיל) לפונקציה?
 3. איך נדע לאיזו בתובת לחזור בסיום הפונקציה?
 4. איך אפשר להעביר פרמטרים לפונקציה?
 5. איפה נשמר משתנים מקומיים (локליים)?
 6. איך אפשר להעביר ערכי חזרה?

מבנה של פונקציה



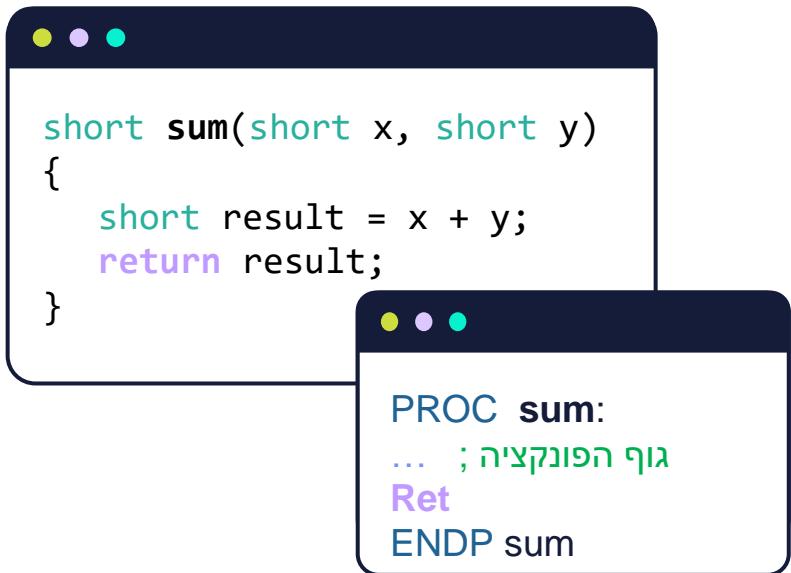
בצורה הבסיסית, כל מה שצריך לעשות על מנת להגדיר פונקציה הוא:

- ו. לחת לה שם (לדוגמא ע"י **תוויות**)
- ו. לבתוב את הפקודות בגוף הפקציה
- ו. לסיים עם פקודה **RET** (ナルם עליה בהמשך)



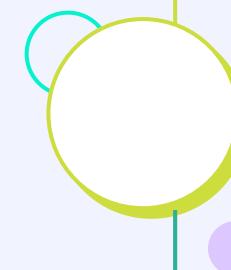
חשוב!

מבנה של פונקציה



קיימת צורת כתיבה נוספת (שකולה לחלווטין
לקודמת)
משתמשת בהוראות אסמלר **ENDP-PROC** ו-
היתרון היחיד שלה: הופך את הקוד לקריא יותר
למDEBUGGING

נשתמש בצורה זו לכתיבה הפונקציות בקורס!



פונקציות ופרוצדורות

```
short sum(short x, short y)
{
    short result = x + y;
    return result;
}
```

- לפני כן אנו צריכים לדע� תשובה למספר שאלות:
- .1. מהו המבנה של פונקציה?
 - .2. איך ניתן לקרוא (להפעיל) לפונקציה?
 - .3. איך נדע לאיזו בתובת לחזור בסיום הפונקציה?
 - .4. איך אפשר להעביר פרמטרים לפונקציה?
 - .5. איך נשמר משתנים מקומיים (локליים)?
 - .6. איך אפשר להעביר ערכי חזרה?

בציד נקרא לפונקציה?

```
PROC sum:  
...  
    גוף הפונקציה ;  
    ...  
    Ret  
ENDP sum
```



```
PROC sum:  
...  
    גוף הפונקציה ;  
    ...  
    Ret  
ENDP sum  
  
...  
jmp sum ; sum ;  
לפקודה הראשונה בגוף  
הפונקציה
```

מה דעתכם?
דברו



```
short sum(short x, short y)
{
    short result = x + y;
    return result;
}
sum(1,2);
```

PROC sum:
...
גוף הפונקציה ; ...
Ret
ENDP sum
...
call sum

קריאה לפונקציה

עם זאת, קיימת פקודה ייעודית לקרוא
של פונקציה: פקודה **CALL**

- פקודה זו מבצעת שתי פעולות:
1. דוחפת את הכתובת של הפקודה הבאה
למחסנית (כמו זו **push**)
2. קופצת לכתובת התווית (כמו **sum** **jmp sum**)

פונקציות ופרוצדורות

```
short sum(short x, short y)
{
    short result = x + y;
    return result;
}
```

- לפני כן אנו צריכים לדעת תשובות למספר שאלות:
- .1. מהו המבנה של פונקציה?
 - .2. איך ניתן לקרוא (להפעיל) לפונקציה?
 - .3. איך נדע לאיזו בתובת לחזור בסיום הפונקציה?
 - .4. איך אפשר להעביר פרמטרים לפונקציה?
 - .5. איך נשמר **משתנים מקומיים** (локליים)?
 - .6. איך אפשר להעביר ערכי חזרה?



כתבת חוזה

- ראיינו שבקリアה לפונקציה בעזרת הפקודה **CALL**, אנו נשמר את הכתובת של הפקודה הבאה במחסנית ביציאה מהfonkzia, נשלוף את הכתובת ונקבע אליה

איך נעשה זאת?



```
short sum(short x, short y)
{
    short result = x + y;
    return result;
}
sum(1,2);
```

PROC sum:
...
גוף הפונקציה ; ...
Ret
ENDP sum
...
call sum

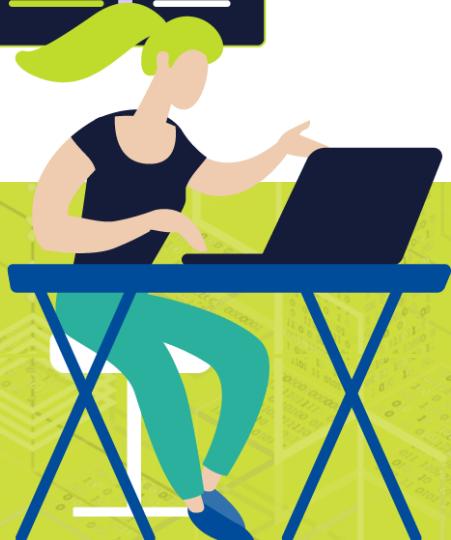
קריאה לפונקציה

פקודה **RET** מבצעת שתי פעולה:

1. שולפת מהמחסנית את כתובות החזרה לאוגר IP (במו דן **pop**)
2. קופצת לכתובת של האוגר IP (במו **jmp ip**)

Live Coding

פָתִיחה בְעַוְרֵךְ קּוֹד ↶



להוסיף תיאור



פונקציות ופרוצדורות

לפני כן אנו צריכים לדעת תשובות למספר שאלות:

```
short sum(short x, short y)
{
    short result = x + y;
    return result;
}
```

- .1 מהו המבנה של פונקציה?
- .2 איך ניתן לקרוא (להפעיל) לפונקציה?
- .3 איך נדע לאיזו כתובת לחזור בסיום הפונקציה?
- .4 איך אפשר להעביר פרמטרים לפונקציה?
- .5 איך נשמר משתנים מקומיים (локליים)?
- .6 איך אפשר להעביר ערכי חזרה?

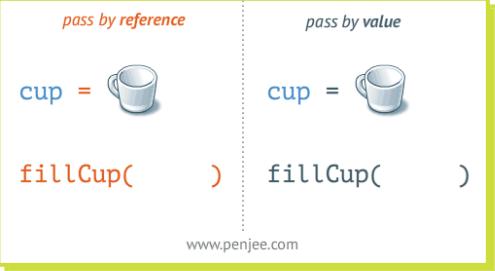
פונקציה – העברת פרמטרים

ראינו איך ניתן לקרוא לפונקציה, אך איך נוכל להעביר לה פרמטרים?

```
short sum(short x, short y)
{
    short result = x + y;
    return result;
}
```

מה דעתכם?
דברו

פונקציה – העברת פרמטרים



- ניתן להעביר פרמטרים בשתי דרכים:
 - באמצעות האוגרים
 - באמצעות המחסנית
- העברת משתנים **By Reference** לעומת **By Value** לעומת:
 - האם נעביר את הערך עצמו או את המצביע לכתובת?
 - מה ההבדל?



פונקציה – העברת פרמטרים

העברת פרמטרים – באמצעות אוגרים

לפני הפקודה **CALL**, נשמר את הפרמטרים באוגרים (לדוגמא ע"י הפקודה 5, ax **mov**), ובגוף הפונקציה נשלוף את הפרמטרים מהאוגרים

בעיה אפשרית: מהי כמות המקסימאלית של פרמטרים שנוכל להעביר בשיטה זו? !

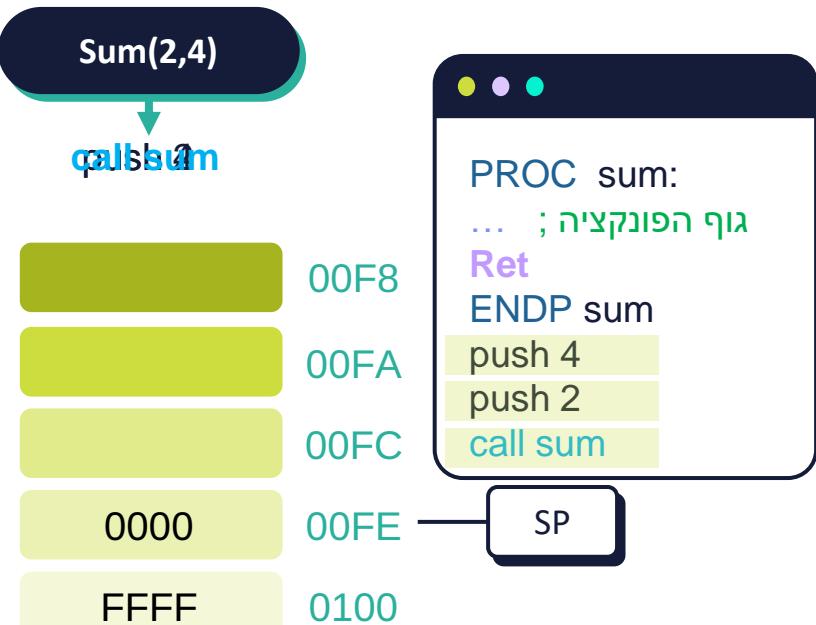
ההעברה פרמטרים – באמצעות המחסנית

לפני הפקודה **CALL**, נשמר את הפרמטרים במחסנית בעזרת פקודת **PUSH**

- נראה בשקף הבא דוגמא להעברה בשיטה זו

העברה פרמטרים

לחצו לビיצו הפקודה



הפקודה (2,4) מתרגם ל:

Push 4
Push 2
Call sum

פונקציות ופרוצדורות

לפני כן אנו צריכים לדעת תשובה למספר שאלות:

```
short sum(short x, short y)
{
    short result = x + y;
    return result;
}
```

- .1 מהו המבנה של פונקציה?
- .2 איך ניתן לקרוא (להפעיל) לפונקציה?
- .3 איך נדע לאיזו בתובת לחזור בסיסום הפונקציה?
- .4 איך אפשר להעביר פרמטרים לפונקציה?
- .5 **איפה נשמר משתנים מקומיים (локליים)?**
- .6 איך אפשר להעביר ערכי חזרה?



פונקציה – משתנים מקומיים

```
short sum(short x, short y)
{
    short result = x + y;
    return result;
}
```

היכן נשמר משתנים מקומיים? **על המחשבנית!**

פונקציה – משתנים מקומיים

בתחילת כל פונקציה אנחנו "נקצה" מקום לכל המשתנים המקומיים של הפונקציה:

- נחשב את כמות הבטים הדרושים לאחסן את כל המשתנים המקומיים (בדוגמא שלנו – 2 בטים)
- נקזה זיכרון על המחסנית ע"י הזאת מצביע המחסנית (לפי כמות הבטים שהיחסנו מעל):
 sub sp, 2

פונקציות ופרוצדורות

- לפני כן אנו צריכים לדעת תשובות למספר שאלות:
- .1. מהו המבנה של פונקציה?
 - .2. איך ניתן לקרוא (להפעיל) לפונקציה?
 - .3. איך נדע לאיזו בתובת לחזור בסיום הפונקציה?
 - .4. איך אפשר להעביר פרמטרים לפונקציה?
 - .5. איך נשמר משתנים מקומיים (локליים)?
 - .6. איך אפשר להעביר ערכי חזרה?

```
short sum(short x, short y)
{
    short result = x + y;
    return result;
}
```



פונקציה – העברת ערך חוזרת

ההעברה ערך חוזרת דומה להעברת פרמטרים לפונקציה, עם הבדל קטן:

יש לבלי היותר ערך חוזרת אחד בלבד. נשמע מוכר?!

לכן נהוג להעביר ערך חוזרת באוגר AX

פונקציות ופרוצדורות

- לפני כן אנו צריכים לדעת תשובות למספר שאלות:
- .1. מהו המבנה של פונקציה?
 - .2. איך ניתן לקרוא (להפעיל) לפונקציה?
 - .3. איך נדע לאיזו בתובת לחזור בסיסום הפונקציה?
 - .4. איך אפשר להעביר פרמטרים לפונקציה?
 - .5. איך נשמר משתנים מקומיים (локליים)?
 - .6. איך אפשר להעביר ערכי חזרה?

```
short sum(short x, short y)
{
    short result = x + y;
    return result;
}
```

שיעור 4

מחסנית ופונקציות



סיכום

מסגרת

מבנה
פונקציה

פעולות
המחסנית

מבנה
המחסנית

יצירה וקיפול של מסגרת

```
• • •  
;create frame  
push bp  
mov bp,sp  
  
;function code  
  
;close frame  
mov sp,bp  
pop bp
```

- יצרת מסגרת מכילה שנייה שלבים
 - .1. שמירת ערך המסגרת הקודם
 - .2. הגדרת ערך של BP בנק' יחוּס לערך של SP בתחילת הפונקציה

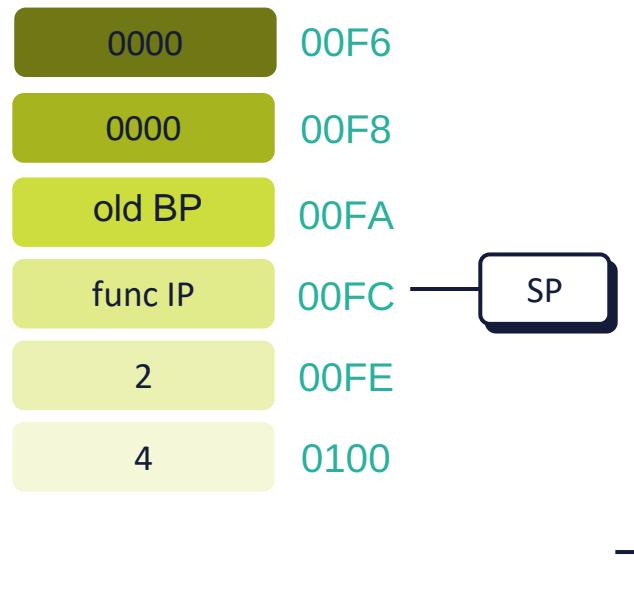
- קיפול המסגרת מכיל את השלבים ההפוכים
 - .1. החזרת ערך SP
 - .2. שחזור BP

יצירה וקיפול של מסגרת

הפעל ◀

```
push bp  
mov bp,sp  
sub sp,4  
;function code  
  
mov sp,bp  
pop bp
```

- נסתכל על המחסנית ביחד לBP ומה היתרונו המשמעותי שלו



יצירה וקיפול של מסגרת

הפעל ◀

```
push bp  
mov bp,sp  
sub sp,4  
;function code  
  
mov sp,bp  
pop bp
```

- נסתכל על המחסנית ביחד לBP ומה היתרונו המשמעותי שלו

| | | |
|---------|------|----|
| 0000 | 00F6 | SP |
| 0000 | 00F8 | |
| old BP | 00FA | BP |
| func IP | 00FC | |
| 2 | 00FE | |
| 4 | 0100 | |

- אפשר לחלק בעט את המחסנית לשנים פרטניים אשר נמצאים בתכובות גבוהות זהה של BP ומשתנים מקומיים בכוכבות נמוכות



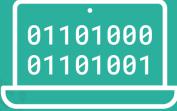
השלם את הקוד

| | | |
|---------|------|------|
| var2 | 00F6 | → SP |
| var1 | 00F8 | |
| old BP | 00FA | → BP |
| func IP | 00FC | |
| parm1 | 00FE | |
| parm2 | 0100 | |

```
• מה הערך של  
parm1   ○  
var1   ○  
parm2   ○  
var2   ○  
.
```

```
mov ax, [bp+parm1]  
mov ax, [bp+var1]  
mov ax, [bp+parm2]  
mov ax, [bp+var2]
```

ארQUITקטורה
אסmbלי



ביצוע Hands-On

*נמצא בתיקית החניכים



סיקום שיעור



תרגיל הבית



מעבר נוסף
על המציגת



איך כדי לחזור
על החומר?

שאלות? תהיה?

תכתבו עבשו שאלות בציג

@



@

ארקיטקטורה
אסמבלי**מג'זינאים**
תכנית הסייבר הלאומית**המרכז לחינוך סייבר**
CYBER EDUCATION CENTER

שיעור 4

סיימנו!!!

סיכום

מסגרת

מבנה פונקצייה

פעולות
המחסניתמבנה
המחסנית