פרק 9
מחסנית ופרוצדורות
נושאים מתקדמים

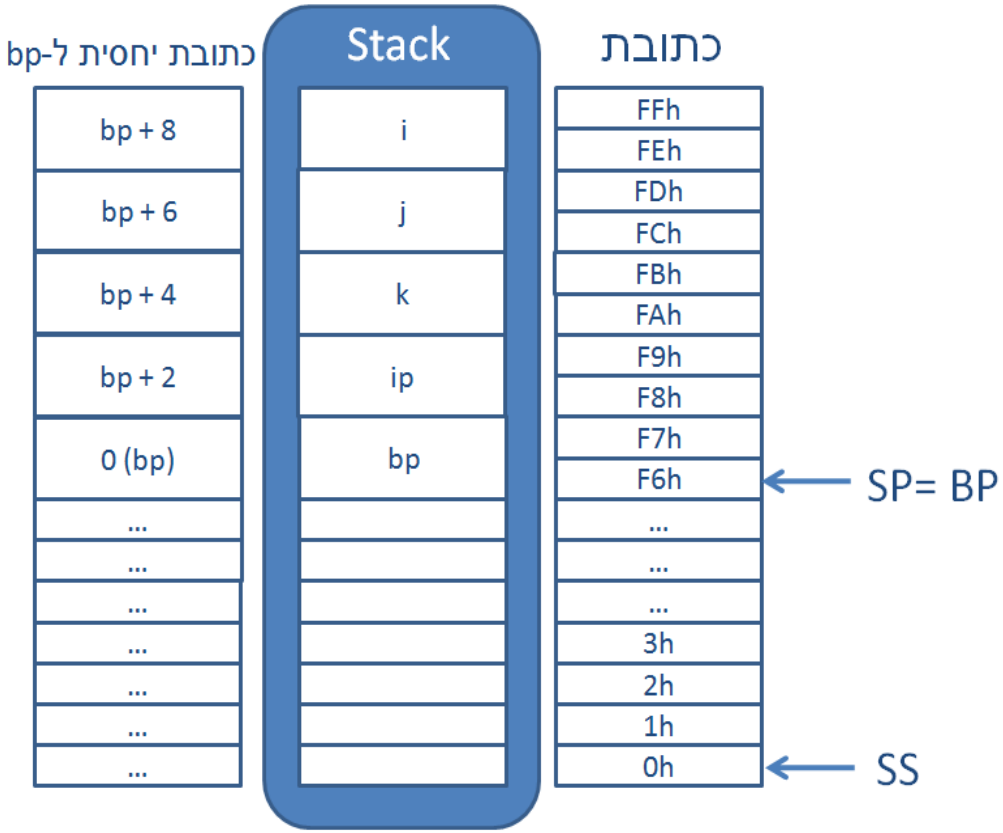
ברק גונן

שימוש ב-bp

```
proc    SimpleProc
        push    bp
        mov     bp, sp
        ; Procedure code
        …
        pop     bp
        ret     6
endp    SimpleProc
```

- עד עכשיו, כל פעם שמרנו את כתובת החזרה ודחפנו למחסנית.
  אנחנו זונחים את השיטה הזו.
- נלמד את השיטה המקובלת לתכנות אסמבלי
- הרגיסטר bp- base pointer – עוזר לגשת לערכים במחסנית
- נראה איך ניתן לכתוב את SimpleProc בדרך שונה
  - נעבור אחד אחד על כל החלקים החדשים בתוכנית (מסומנים)

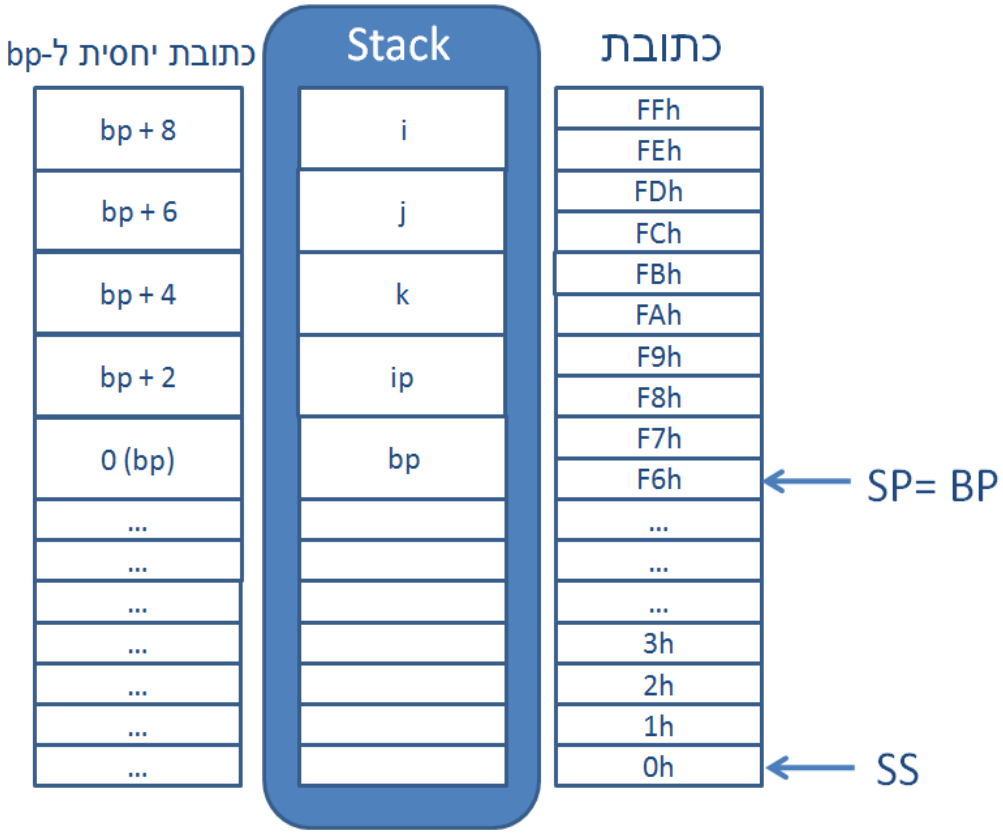
שימוש ב-bp - המשך




```
push    bp
mov     bp, sp
```

- תחילה שומרים את ערכו של bp, כדי שנוכל לשחזר את ערכו ביציאה
- מצב המחסנית בתוך הפרוצדורה, לאחר פקודת ההעתקה של sp לתוך bp
  - שימו לב לכתובת היחסית ל-bp
  - מה יצא לנו מזה?

שימוש ב-bp - המשך



- ערכו של bp נשאר קבוע משך חיי הפרוצדורה
- bp מצביע תמיד על אותו מקום במחסנית
- אפשר להתייחס לכל מיקום במחסנית יחסית למיקום של ש-bp מצביע עליו

שימוש ב-bp- המשך

```
proc    SimpleProc
        push    bp
        mov     bp, sp
        ; Compute I+J-K
        xor     ax, ax
        add     ax, [bp+8]
        add     ax, [bp+6]
        sub     ax, [bp+4]
        pop     bp
        ret     6
endp    SimpleProc
```

- נראה איך אפשר להשתמש במסקנה האחרונה:
  - [bp+4] תמיד מצביע על K
  - [bp+6] תמיד מצביע על J
  - [bp+8] תמיד מצביע על I

הוראת EQU

- מורה לאסבמלר להחליף צירוף תווים מוגדר בצירוף תווים אחר
  - iParm equ [bp+8]
  - kParm equ [bp+6]
  - jParm equ [bp+4]
- כיוון ש-EQU היא הוראה לאסמבלר, כמו תוויות, היא אינה תופסת מקום בזיכרון
  - התווים מוחלפים בתווים אחרים בשלב האסמבלר
- מומלץ להשתמש כאשר מבצעים pass by value
- בשיטת pass by reference ממילא משתמשים ברגיסטר כגון bx

פרוצדורה- לפני ואחרי

- אותו קוד- לפני ואחרי:

```
proc    SimpleProc
        pop     ReturnAddress
        ; Compute I+J-K
        pop     ax          ; k
        pop     bx          ; j
        sub     bx, ax      ; j-k
        pop     ax          ; i
        add     ax, bx      ; i+j-k
        push    ReturnAddress
        ret
endp    SimpleProc
```

```
proc    SimpleProc
        push    bp
        mov     bp, sp
        ; Compute I+J-K
        xor     ax, ax
        add     ax, iParm
        add     ax, jParm
        sub     ax, kParm
        pop     bp
        ret     6
endp    SimpleProc
```

- נוסף על כך שהקוד יותר קריא, חסכנו רגיסטר ואנחנו יכולים לעשות שימוש חוזר בפרמטרים

פקודת ret עם קבוע

- נותר לנו להבין מה משמעות הפקודה ret 6
- כזכור פקודת ret:
  - מוציאה את כתובת החזרה מהמחסנית
  - מעלה את ערכו של sp ב־2
  - קופצת אל כתובת החזרה
- פקודת ret עם קבוע:
  - מוציאה את כתובת החזרה מהמחסנית
  - מעלה את ערכו של sp ב־2 **פלוס הקבוע**
  - קופצת אל כתובת החזרה
- בצורה זו ניתן לנקות מהמחסנית פרמטרים ששלחנו

יתרונות השימוש ב-bp- סיכום ביניים

- בתחילת הפרוצדורה לא צריך לעשות pop לכתובת החזרה ולשמור אותה.
- לא צריך לעשות pop לכל הערכים שדחפנו למחסנית.
  פשוט ניגשים ישר אל הכתובת שלהם במחסנית בעזרת bp.
- אפשר לייצג כל תא בזיכרון המחסנית על־ידי שם קריא ובעל משמעות.
- יתרון נוסף של bp, שנראה אותו מיד, הוא ש-bp עוזר לנו לגשת למשתנים מקומיים.

תרגיל- גישה לפרמטרים ע"י bp

- כיתבו פרוצדורה Pitagoras. הפרוצדורה תקבל 3 פרמטרים- X, Y, Z ותבדוק אם מתקיים היחס $X^2+Y^2=Z^2$. אם כן- הפרוצדורה תקבע $ax=1$, אחרת $ax=0$.

שימוש במחסנית להגדרת משתנים מקומיים

- משתנה מקומי:
  - Local Variable
  - אינו מוכר מחוץ לפרוצדורה
  - מוגדר ע"י הפרוצדורה
    - הפרוצדורה מקצה זיכרון על המחסנית
    - משחררת את הזיכרון לפני החזרה לתוכנית
    - איך זה מתבצע בפועל?

הגדרת משתנים מקומיים

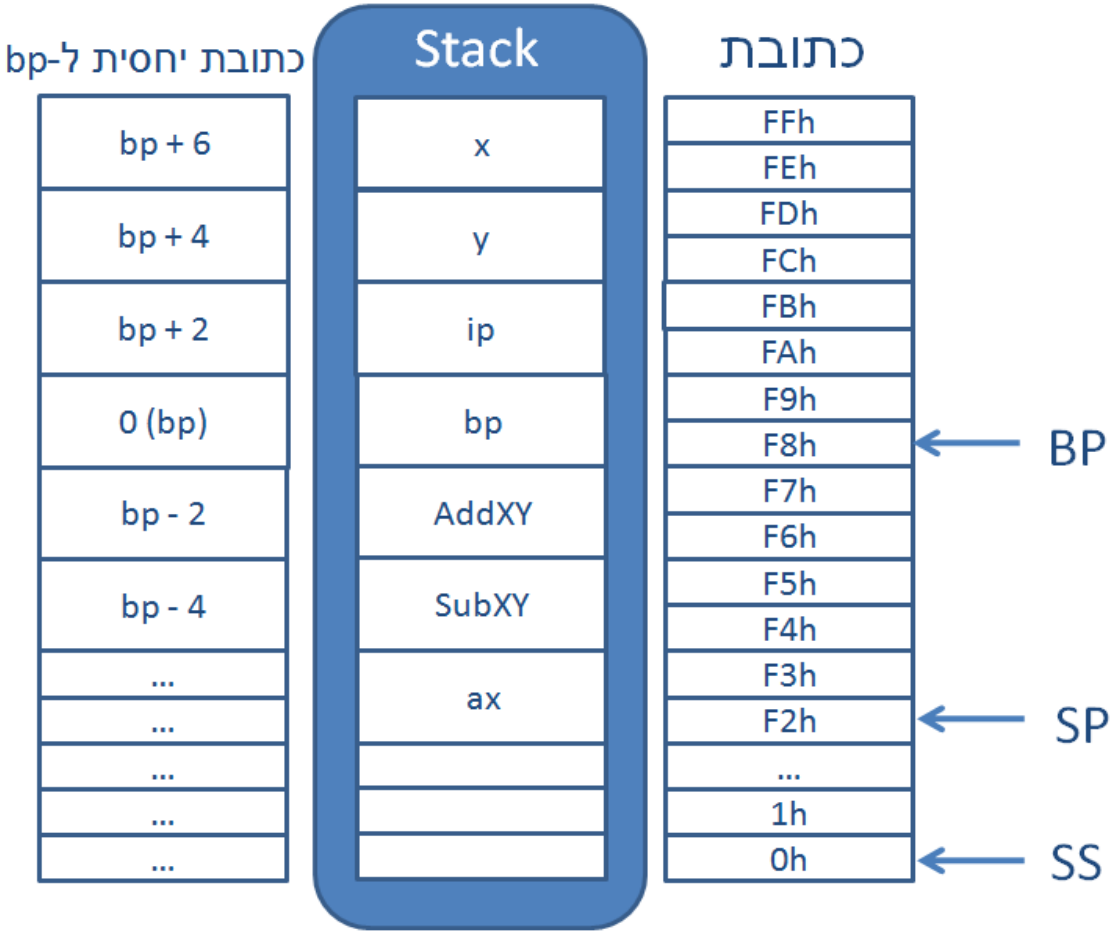
- ראינו שכל הקצאת זיכרון על המחסנית מורידה את ערכו של sp
- כדי להקצות משתנים מקומיים- מורידים את ערכו של sp
  - לדוגמה: הקצאת 6 בתים למשתנים מקומיים-
  - Sub sp, 6
- כדי לשחרר את הזיכרון – ולהביא את sp כך שיצביע על המקום במחסנית בו שמור ip- מבצעים את הפעולה ההפוכה
  - לדוגמה: שחרור 6 בתים שהוקצו-
  - Add sp, 6

הגדרת משתנים מקומיים- דוגמה

```
varX    equ     [bp+6]
varY    equ     [bp+4]
AddXY   equ     [bp-2]
SubXY   equ     [bp-4]
proc    XY
        push    bp
        mov     bp, sp
        sub     sp, 4       ; Allocate
        push    ax          ; Save ax
        mov     ax, varX
        add     ax, varY
        mov     AddXY, ax
        mov     ax, varX
        sub     ax, vary
        mov     SubXY, ax
        pop     ax          ; Restore ax
        add     sp, 4       ; De allocate
        pop     bp
        ret     4
endp    XY
```

- נגדיר פרוצדורה בשם XY
  - מקבלת שני פרמטרים- x, y
  - מגדירה שני משתנים מקומיים- AddXY, SubXY
  - AddXY- סכום הפרמטרים
  - SubXY- הפרש הפרמטרים

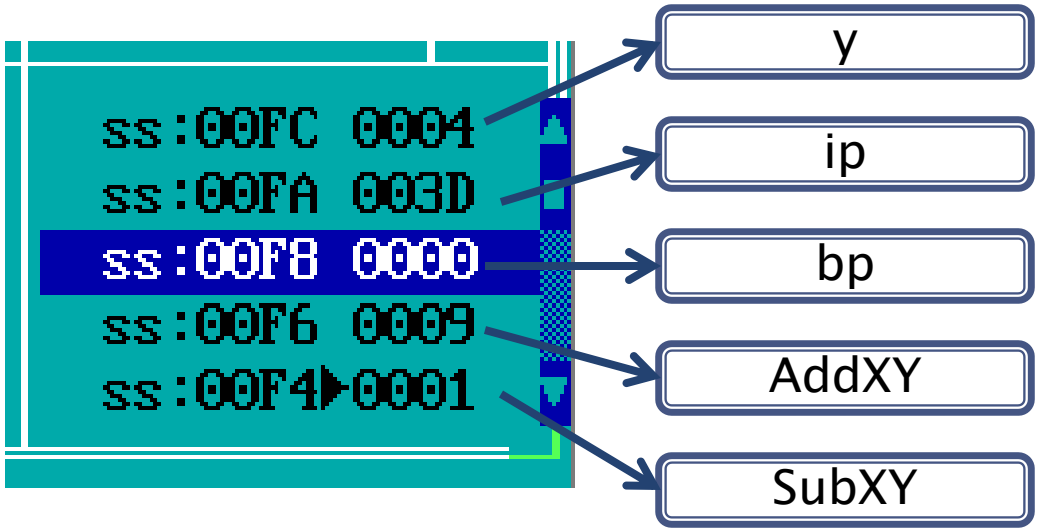
פרוצדורה XY- מצב המחסנית



פרוצדורה XY- המשך

```
mov     [x], 5
mov     [y], 4
push    [x]
push    [y]
call    XY
```

- נקרא לפרוצדורה XY מהתכנית הראשית:
- כך נראית המחסנית:



תרגיל- משתנים מקומיים

- כיתבו תוכנית ראשית, שדוחפת למחסנית שלושה ערכים כלשהם וקוראת לפרוצדורה בשם XYZ. הפרוצדורה כוללת שלושה משתנים מקומיים – LocalX, LocalY, LocalZ. הפרוצדורה תעתיק כל ערך לתוך משתנה מקומי אחר. הריצו את התוכנית ובידקו במחסנית שההעתקה בוצעה כנדרש.

שימוש במחסנית להעברת מערך לפרוצדורה

- לעיתים נרצה להעביר לפרוצדורה אוסף של איברים הנתונים במערך
  - העברה בשיטת pass by value:
    - דחיפת האיברים אחד אחד... מייגע!
  - בשיטת pass by reference מעבירים רק:
    - כתובת האיבר הראשון במערך
    - מספר האיברים במערך
    - בתוך הפרוצדורה, ניגשים לאלמנט במערך ע"י כתובת הבסיס וההיסט

```
DATASEG
num_elements    equ     15
Array   db      num_elements
                dup (?)

CODESEG
push    num_elements
push    offset  Array
call    SomeProcedure
```

תרגילים: העברת מערכים לפרוצדורות

- צרו פרוצדורה שמקבלת מערך ואת המשתנה sum ומכניסה לתוך sum את סכום האיברים במערך. לדוגמה, עבור המערך 2,2,3,4,5 התוצאה תהיה sum=16.

תרגילים: העברת מערכים לפרוצדורות

- כיתבו פרוצדורה SortArray שמקבלת מצביע למערך ומספר איברים במערך, וממיינת את המערך מהאיבר הקטן לגדול. לדוגמה עבור המערך 3,6,5,2,1 הפרוצדורה תגרום למערך להכיל את הערכים: 1,2,3,5,6. הדרכה לכתיבת הפרוצדורה:
  - כיתבו פרוצדורת עזר FindMin שמקבלת מצביע למערך, מוצאת את האיבר הקטן ביותר ומחזירה את האינדקס שלו.
  - כיתבו פרוצדורת עזר Swap שמקבלת שני פרמטרים באמצעות שיטת pass by reference ומחליפה את הערכים שלהם.
  - הפרוצדורה SortArray תרוץ בלולאה על המערך ותקרא ל־FindMin. לאחר מכן תקרא ל־Swap עם שני פרמטרים: האינדקס שהחזירה FindMin והאינדקס הראשון במערך.
  - לאחר שהפרוצדורה SortArray העבירה את האיבר הקטן ביותר לאינדקס הראשון, היא תקרא ל־FindMin כאשר המצביע למערך הוא על האינדקס השני במערך. לאחר מכן, SortArray תקרא ל-Swap עם שני פרמטרים: האינדקס שהחזירה FindMin והאינדקס השני במערך.
  - הפרוצדורה תמשיך בשיטה זו עד לסיום מיון המערך.
  - בתוכנית הראשית: השתמשו בפקודות קלט ופלט כדי לקלוט איברים למערך וכדי להדפיס את המערך הממוין. הניחו שהמספרים הם בני ספרה אחת.
    - אתגר: הניחו מספרים בני יותר מספרה אחת.

תרגילים: העברת מערכים לפרוצדורות

- כיתבו פרוצדורה שנקראת Sort2Arrays, שמקבלת מצביעים לשני מערכים ומספר איברים בכל מערך, ומצביע נוסף למערך יעד sorted אליו יש להכניס את תוצאת המיון של שני המערכים, כאשר ערכים כפולים מסוננים החוצה. לדוגמה:
  - Array1 = 4,9,5,3,2
  - Array2 = 3,6,4,1
- לאחר הרצת הפרוצדורה:
- Sorted = 1,2,3,4,5,6,9
- הדרכה לכתיבת הפרוצדורה:
  - כיתבו פרוצדורה בשם Merge שמקבלת מצביעים לשני מערכים ומעתיקה אותם למערך אחד, ללא מיון או סינון.
  - קיראו ל־SortArray עם המערך שיצרה Merge.
  - כיתבו פרוצדורה בשם Filter שעוברת על המערך הממוין ומאפסת את כל הערכים שמופיעים יותר מפעם אחת.

Stack Overflow

- Stack Overflow הינו מונח חשוב מתחום אבטחת המידע
  - ניתן להריץ קוד שונה ממה שתוכנן
  - איננו מעודדים שינוי קוד מקור- זה אינו חוקי!
- נעסוק בתיאוריה של הנושא:
  - מודעות לבעיות אבטחה היא כלי חשוב לעתידכם
  - הבנה של הנושא דורשת חזרה והתעמקות בחומר הלימוד

Stack Overflow

- **Buffer Overflow-** מגדירים קטע זיכרון (באפר), ומנסים להעתיק אליו יותר בתים מהגודל שלו.

ה"באפר" הוא בגודל 12 ביצים. אם ננסה להכניס 13 ביצים נקבל Buffer Overflow.

- אם הבאפר שגלש הוגדר על המחסנית, הגלישה נקראת **Stack Overflow**

Stack Overflow- המשך

- בררו לעצמכם- מה עושה התוכנית הבאה?
- www.cyber.org.il/assembly/stackof.asm
- התוכנית מדפיסה למסך בקשה לקלוט את שם המשתמש. לאחר שהמשתמש סיים עליו להקיש enter. אז התכנית מדפיסה למסך הודעה Program finished. כך:

```
Please enter your name, press enter to finish
Jon Snow

Program finished
```

Stack Overflow- המשך

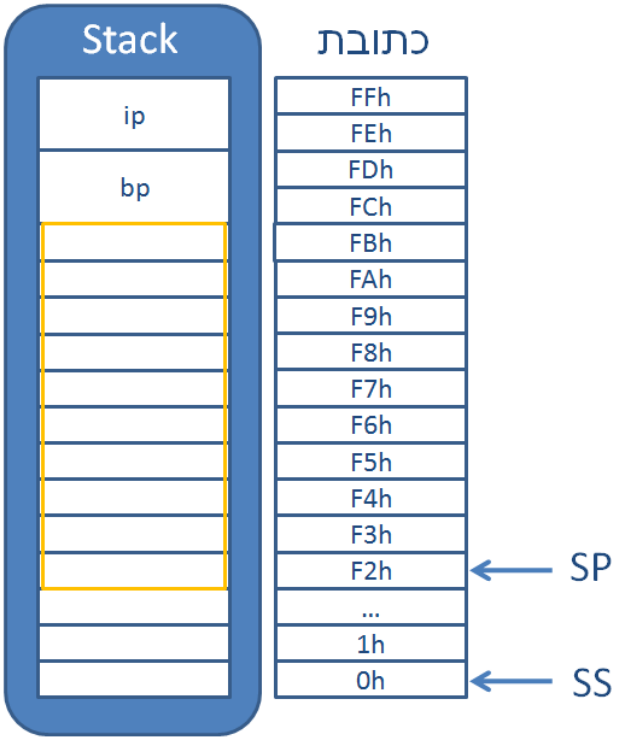
- לאחר קטע הקוד שמסיים את ריצת התוכנית, יש קטע קוד נוסף, שמדפיס הודעה שונה למסך.
  - זהו קטע קוד מוזר, מכיוון שהוא לא אמור להיות מורץ.
  - אין בתכנית שום פעולה שמקפיצה את ip כך שיגיע אל קטע הקוד הזה.
  - אם כך, איך אפשר להסביר את ההרצה הבאה?

```
Please enter your name, press enter to finish
Jon Snow     !

Here be dragons
```

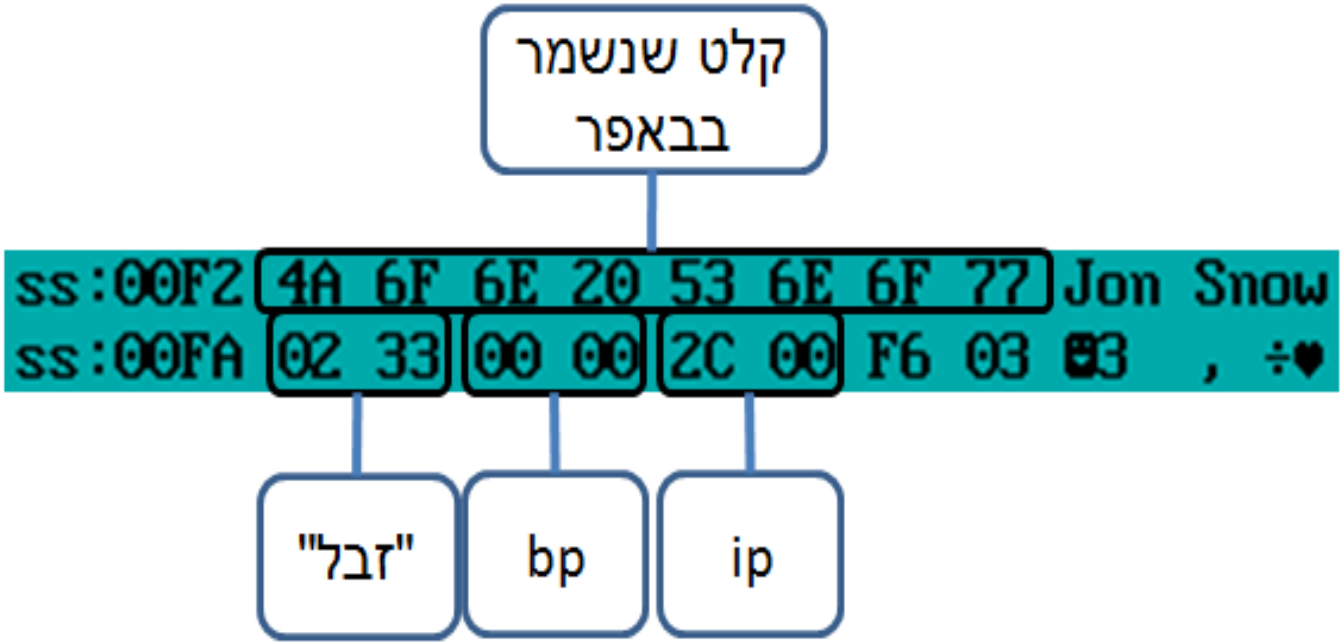
Stack Overflow- המשך

- הפרוצדורה GetName מגדירה באפר בגודל 10 בתים על המחסנית. בתוך הפרוצדורה, המחסנית נראית כך:



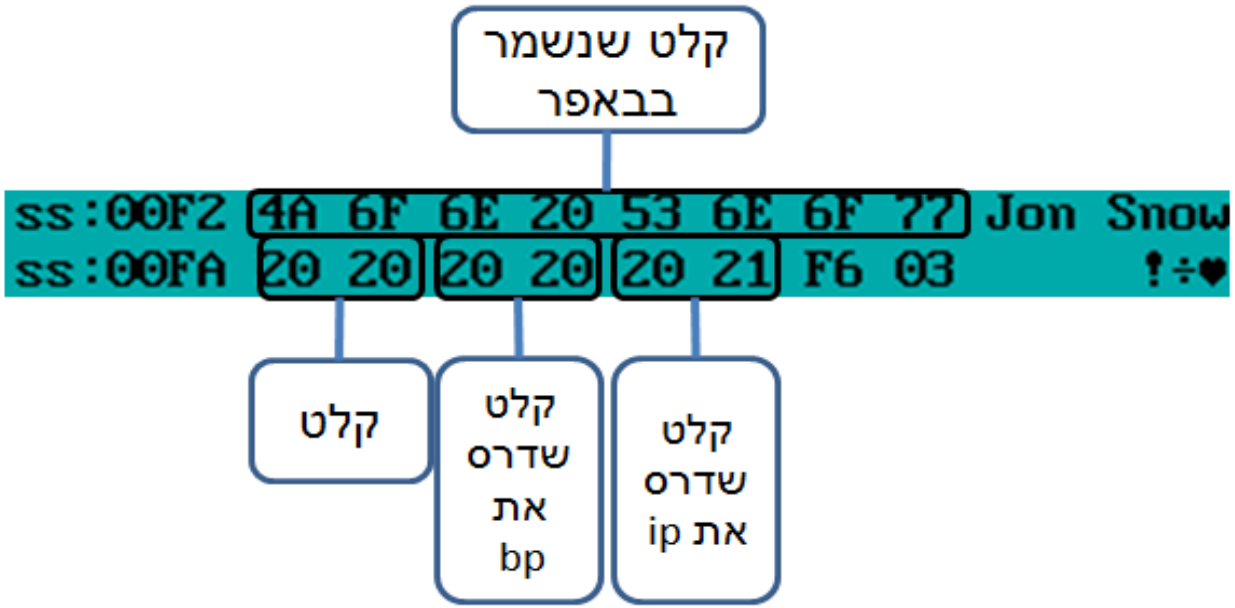
Stack Overflow- המשך

- בהרצה הראשונה המשתמש הזין את שמו. מצב המחסנית:



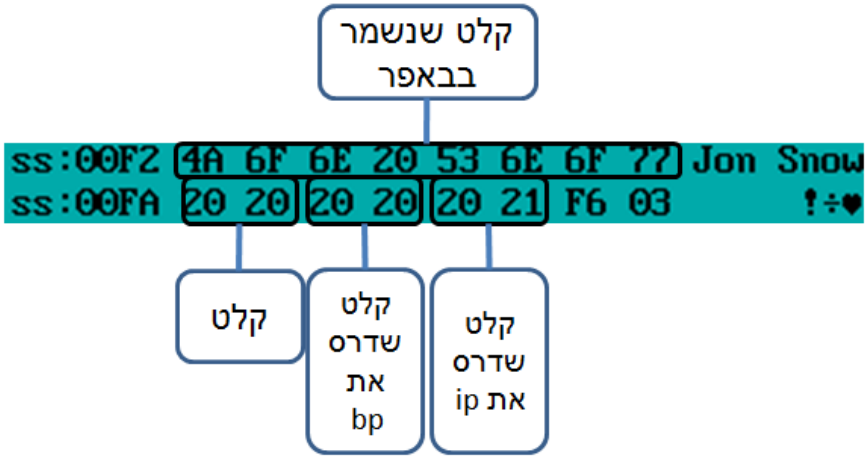
Stack Overflow- המשך

- נוסף על 8 התווים הראשונים המשתמש מזין 5 תווי רווח ואז סימן קריאה.
  - כתוצאה מהעתקה של 14 בתים לתוך באפר שגודלו 10 בתים, נוצר מצב של Stack Overflow.



Stack Overflow- המשך

- התוכנית ממשיכה לרוץ- תעצור רק אם תיתקל ב-opcode לא חוקי
- בסיום ריצת הפרוצדורה:
  - הבאפר נמחק (פקודת add sp, 10)
  - מבוצע pop bp, אך הערך הוא כעת 2020h
  - מבוצע ret לכתובת 2120h (לא 2021h- little endian)



Stack Overflow- המשך

מפת Psalter מ-1265. בתחתית המפה, מחוץ לטריטוריה המוכרת לאדם, דרקונים.

- פעולת ה-ret מקפיצה את התוכנית לכתובת 2120h שם נמצא הקוד שמדפיס את הודעת הסיום 'Here be dragons'
  - הביטוי לקוח ממפות עתיקות ופירושו: איזור לא מוכר, שאין לדעת מה נמצא בו. צפו לסכנות...
- איך אפשר לשנות את התוכנית כך שיימנע ה-Stack Overflow?

בחינות בגרות: פרוצדורות

- מספר שאלון 899205
  - קיץ 2014
  - קיץ 2011
  - קיץ 2010

קיץ 2014

8. מערך חד־ממדי ARR בגודל N של מספרים שלמים — כאשר N זוגי — ייקרא **מאוזן** אם:
הסכום של הערך המאוחסן בתא הראשון של המערך ושל הערך המאוחסן בתא האחרון של המערך שווה לסכום של הערך המאוחסן בתא השני של המערך ושל הערך המאוחסן בתא שלפני האחרון במערך, ושווה לסכום של הערך המאוחסן בתא השלישי של המערך ושל הערך המאוחסן בתא השלישי מסוף המערך, וכך הלאה. וסכום זה שווה לערך המאוחסן במשתנה x.

<u>לדוגמה</u>:

בעבור מערך ARR בגודל 6 ומשתנה x שהערך המאוחסן בו הוא 10,

	0	1	2	3	4	5
ARR	3	1	4	6	9	7

המערך ARR **מאוזן** כי מתקיים: $10 = 4 + 6 = 1 + 9 = 3 + 7$.

א. כתוב באסמבלר שגרה (פרוצדורה) בשם CHECK, שתקבל כפרמטרים שלוש כתובות של תאים בזיכרון. השגרה תבדוק אם סכום הערכים המאוחסנים בתאים שבשתי הכתובות הראשונות שווה לערך המאוחסן בתא שבכתובת השלישית. אם כן — השגרה תאחסן 1 באוגר DL, אחרת — היא תאחסן בו 0.

ב. במקטע הנתונים מוגדרים מערך ARR ומשתנה x:

```
ARR     DB 100     DUP   (?)
X       DB ?
```

כתוב באסמבלר קטע תכנית שיבדוק אם המערך ARR הוא **מאוזן**.
אם כן — קטע התכנית יאחסן 1 באוגר DH, אחרת — הוא יאחסן בו 0.
הנח שסכום הערכים של שני תאים <u>אינו</u> עולה על 255.
עליך להשתמש בשגרה CHECK שכתבת בסעיף א.

קיץ 2011

6. מערך A נקרא **מוכל** במערך B , אם האורך של מערך A אינו עולה על האורך של מערך B, וכל הערכים של איברי מערך A נמצאים במערך B ברצף ובאותו סדר כמו במערך A .

בכל מערך מאוחסנים מספרים שלמים עם סימן, השונים זה מזה.

לדוגמה:

בעבור המערכים A ו- B שלפניך, המערך A **מוכל** במערך B .

מערך A:

0	1	2	3	4	5
3	−5	0	8	5	11

מערך B:

0	1	2	3	4	5	6	7	8	9
15	−7	4	3	−5	0	8	5	11	2

איברי המערך B המסומנים באפור הם הערכים של איברי המערך A .

א. במקטע הנתונים הוגדרו הנתונים כך:

```
ARR_B   DB  100   DUP (?)
ARR_A   DB  10    DUP (?)
V       DB  ?
P       DB  ?
```

כתוב באסמבלר שגרה (פרוצדורה) בשם TEST , שתקבל ערך שמאוחסן במשתנה V .
השגרה תבדוק אם הערך שמאוחסן במשתנה V נמצא במערך ARR_B .
אם כן – השגרה תאחסן במשתנה P את האינדקס של האיבר המתאים,
אחרת – השגרה תאחסן במשתנה P את הערך 1− .

ב. כתוב באסמבלר קטע תכנית שיבדוק אם המערך ARR_A **מוכל** במערך ARR_B .
אם כן – קטע התכנית יאחסן 1 באוגר BL , אחרת – הוא יאחסן 0 באוגר BL .
עליך להשתמש בשגרה TEST שכתבת בסעיף א.
הנח שהאורך של המערך ARR_A אינו עולה על האורך של מערך ARR_B .

קיץ 2010

7. נגדיר מערך **ממוין למחצה** באופן הזה:

מערך באורך זוגי המכיל מספרים גדולים מ־ 0 ושונים זה מזה, יש בו רצף של איברים סמוכים ממוין בסדר עולה, ואורך הרצף הזה לפחות מחצית מאורך המערך.

לדוגמה:

המערך בגודל 10 שלפניך הוא **ממוין למחצה**.

הרצף של איברים הממוינים בסדר עולה נמצא בין מציין 2 למציין 6 (מסומן באפור) ואורכו 5.

0	1	2	3	4	5	6	7	8	9
8	4	3	5	10	19	35	7	1	9

במקטע הנתונים הוגדרו הנתונים כך:

```
LOW     DB   ?
HIGH    DB   ?
ARR     DB   100  DUP(?)
```

א. כתוב באסמבלר שגרה (פרוצדורה) בשם CHECK , שתקבל את המערך ARR ושני מספרים שלמים LOW ו־ HIGH (LOW קטן מ־ HIGH). שני המספרים הם מציינים (אינדקסים) של איברים במערך.

השגרה (פרוצדורה) תבדוק אם איברי המערך, מהאיבר שהמציין שלו הוא LOW ועד האיבר שהמציין שלו הוא HIGH (כולל LOW ו־ HIGH), ממוינים בסדר עולה.

אם כן — השגרה (פרוצדורה) תאחסן 1 באוגר AH , אחרת — היא תאחסן 0 ב־ AH.

הנח שאיברי המערך ARR גדולים מ־ 0 ושונים זה מזה.

ב. כתוב באסמבלר קטע תכנית שיבדוק אם המערך ARR הוא מערך **ממוין למחצה**.

אם כן — קטע התכנית יאחסן 1 באוגר AH , אחרת — הוא יאחסן 0 ב־ AH.

עליך להשתמש בשגרה (פרוצדורה) CHECK שכתבת בסעיף א.