הקדמה - הקצאת מקום בזיכרון

בשיעור זה נדבר על הקצאת מקום בזיכרון לטובת שמירת ערכי מידע.
נדבר על מערכים שתלויים בסוג המשתנה אותו נרצה לאחסן בכל פעם,
ונציג שתי שיטות לסידור הבתים של הערכים אותם נשמור.

הקדמה - הקצאת מקום בזיכרון

הזיכרון בנוי מתאים, לכל תא יש כתובת, והוא יכול לאחסן **8 סיביות** כלומר **בית** אחד.
הכתובות בזיכרון הן ביחידות מידה של בתים.
נשים לב שיחידת המידה **מילה** משמעותה היא **32 סיביות**, מכיוון שבכל כתובת ניתן לאחסן 8 סיביות שהן בית אחד, אז כדי לאחסן מילה שלמה נצטרך **4 כתובות** בזיכרון כלומר **4 בתים**.
כאשר נרצה להקצות מקום בזיכרון, עלינו תחילה לומר כמה מקום תתפוס כל יחידת מידע שבה נעסוק, למשל:

- אם יחידת המידע היא **מילה** אז אחרי ההכרזה, בכל פעם יוקצו 4 בתים.
  נקבל מערך של מילים. מערך זה יתחיל בכתובת המתחלקת ב-4.
- אם יחידת המידע היא **חצי מילה** (half word) אז אחרי ההכרזה בכל פעם יוקצו 2 בתים.
  נקבל מערך של חצי מילים. מערך זה יתחיל בכתובת המתחלקת ב-2.
- אם יחידת המידע היא **בית** (byte) אז אחרי ההכרזה בכל פעם תוקצה מילה אחת.

נציג תחילה כיצד מתבצעת ההכרזה וההקצאה, וניתן מספר דוגמות אפשרויות שונות לסוגי נתונים עליהם נוכל להכריז.

הנחיות אחסון - סוגים

- כיצד מצהירים על מערך בזיכרון?

1. תחילה רושמים שם לתווית, למשל :var1. השם הוא כינוי לכתובת מסוימת בזיכרון שבה יתחיל המערך. במקום לרשום את הכתובת הספציפית בכל פעם, נוכל פשוט לרשום את שם התווית (var1)
   - לצורך העשרה והעמקת ההבנה נאמר שהקישור בין שם התווית ובין הכתובת שאותה היא מייצגת מתבצע בשלב אחר של המרת התוכנית ממלל לשפת מכונה ולא נעסוק בו.
2. לאחר מכן רושמים כמה מידע יתפוס כל תא במערך.
3. לאחר מכן רושמים את הערכים עצמם שיאוחסנו במערך, להלן הדגמה:
   - הערה: בזיכרון כל תו (למשל a או b או \n וכו..) נשמר ע"י 8 סיביות – בית אחד.

Address	Value	Binary Value	
var1	0x41	01000001	; 'A'
var1 + 1	0x45	01000101	; 'E'
var1 + 2	0x7F	01111111	; 127
var1 + 3	0xFF	11111111	; -1
var1 + 4	0x0A	00001010	; '\n'

הדגמה לתוכן הזיכרון עבור המערך של var1

```
.DATA
var1:   .BYTE     'A', 'E', 127, -1, '\n'
var2:   .HALF     -10,  0xffff
var3:   .WORD     0x12345678
Var4:   .WORD     0:10
var5:   .FLOAT    12.3, -0.1
var6:   .DOUBLE   1.5e-10
```

הצהרות שונות על מערכים לדוגמה

- byte. הערכים יאוחסנו בבית כל אחד (8 סיביות)
- half. הערכים יאחסנו בחצי מילה
- word. הערכים יאחסנו במילה
- word w:n. אחסון 32 הסיביות שבמילה w שוב ושוב לתוך **n** מילים רצופות בזיכרון
- float. ערכים עם ספרה אחת אחרי הנקודה
- double. ערכים עם 2 ספרות אחרי הנקודה

סוגי הצהרות על כמות המידע שיתפוס כל תא במערך

שיטות סידור הבתים

- כאשר סוג הנתונים שהמערך שומר תופס יותר מבית אחד בזיכרון - נשאלת השאלה איזה בתים לאחסן קודם, לשם כך קיימות שתי אפשרויות:
  - השיטה big endian: בשיטה זו **הבתים** של הסיביות המשמעותיות יותר יאוחסנו קודם.
    למשל עבור הערך 0xF23C שערכו בבינארית הוא 11110010 00111100 הסיביות האדומות שמיוצגות ע"י ספרות ההקסא F2 משמעותיות יותר מהסיביות בתכלת, מכיוון שהן נמצאות **שמאלה יותר** במיקומן, ולכן הסיביות 11110010 יאוחסנו קודם בזיכרון ורק לאחר מכן הסיביות 00111100.
  - השיטה little endian: בשיטה זו **הבתים** של הסיביות הפחות משמעותיות יאוחסנו קודם.
    ולכן הסיביות 00111100 יאוחסנו קודם בזיכרון ורק לאחר מכן הסיביות 11110010.
    נדגים זאת בשקף הבא:
- השיטות מדברות רק על אופן סידור הבתים **בתוך** כל תא במערך, ולא בין תאים שונים.

דוגמה - סידור הבתים בכתובת

נראה כיצד המילה 0x5A5B5C6D שמאוחסנת בזיכרון החל מהכתובת 3000 בזיכרון שתשובץ בתאים הזיכרון ב-2 השיטות: little endian ו big endian כאשר מדובר במערך של **מילים**.
כלומר כאשר כתוב L1: 0x5A5B5C6D.

לפני כן מספר הערות:

- בכל כתובת בזיכרון, מאוחסן **בית אחד** - כלומר **8 סיביות**.
- ספרה בבסיס 16 מיוצגת על ידי 4 סיביות, מכיוון שבכל תא בזיכרון יש 8 סיביות, אז יש מקום ל-2 ספרות הקסא דצימליות, ולכן התוכן של כל תא מיוצג על ידי 2 ספרות הקסא.
- מילה שלמה היא בגודל של 32 סיביות כלומר 4 בתים, נדרשות אם כן 4 כתובות כדי לייצג אותה.
(כזכור בכל כתובת בזיכרון יש 8 סיביות), לכן אם מדובר במילה 0x5A5B5C6D תשובץ כך:

Little endian	
כתובת	ערך
3000	0x6D
3001	0x5C
3002	0x5B
3003	0x5A

Big endian	
כתובת	ערך
3000	0x5A
3001	0x5B
3002	0x5C
3003	0x6D

דוגמה 2 - סידור הבתים בכתובת

יחידה המידע **חצי מילה** היא יחידת מידה שגודלה 2 בתים (סוג מידע זה מסומן ע"י half.)
נראה כיצד מערך של חצאי מילים (כלומר 2 בתים בכל תא במערך) יאחסן את 2 הערכים הבאים:
0x5A5B , 0x5C6D כאשר הכתובת בזיכרון שבה מתחיל המערך היא 3000.
כלומר נדגים את יישום הנחיית האחסון הבאה: VAR1: 0x5A5B , 0x5C6D.
נדגים את השיבוץ בפועל בתאי הזיכרון ב-2 השיטות: little endian ו big endian

Little endian

כתובת	ערך
3000	0x5B
3001	0x5A
3002	0x6D
3003	0x5C

תא ראשון 16 סיביות

תא שני 16 סיביות

הבית **הפחות** משמעותי בתא מאוחסן קודם

הבית **היותר** משמעותי בתא מאוחסן אחר כך

Big endian

כתובת	ערך
3000	0x5A
3001	0x5B
3002	0x5C
3003	0x6D

תא ראשון 16 סיביות

תא שני 16 סיביות

הנחיית האחסון: VAR1: 0x5A5B , 0x5C6D

סוגי ערכים נוספים

- ascii. מקצה רצף של תווים למחרוזת אסקיי
- asciiz. זהה לקודם, אך מוסיף ערך 0\ (null) לציון סיום המחרוזת
- **n** space. מקצה n בתים לא מאותחלים בקטע הנתונים
- **n** align. מיישר את הגדרה המידע הבא אחריה, למיקום בזיכרון שהוא כפולה של n.
  בד"כ נהוג לבחור את הערך n להיות חזקה חיובית של 2.
  סוג הצהרה זו אינה עומדת בפני עצמה, לאחריה יבוא סוג הנתונים כגון byte. וערכים.

תוויות המאוחסנות ברצף

- כאשר בתחילת הקוד, מוגדרות כמה תוויות ברצף, הכוונה היא בד"כ שהכתובות אליהן הן מתייחסות יסמנו כתובות עוקבות בזיכרון, עם מגבלה:
  - אחסון מילים יהיה רק בכתובות שמתחלקות ב-4
  - אחסון של חצאי מילים יהיה רק בכתובות שמתחלקות ב-2

ניתן דוגמה (נניח שמדובר על big endian):

- נניח שהכתובת שהתווית var1 מייצגת בזיכרון היא 0x3000.
  החל מכתובת זו מאוחסנים 5 ערכים עד לכתובת 0x3004.
  כלומר הכתובת הפנויה הבאה תהיה 0x3005 , התווית הבאה מייצגת חצאי מילים, לכן כיוון שכתובת זו לא מתחלקת ב-2, נדלג על תא אחד ו- var2 ייצג את 0x3006.
- בכתובת של var2 יאוחסנו 2 ערכים בגודל של חצי מילה, לכן ייעשה שימוש ב-4 בתים עד לכתובת 0x3009. הכתובת הפנויה הבאה תהיה 0x300A , כיוון שהתווית var3 מייצגת מערך של **מילים** אז נדלג לכתובת הסמוכה המתחלקת ב-4 0x300C.

תווית	כתובת	ערך
var1	0x3000	A
	0x3001	B
	0x3002	127
	0x3003	-1
	0x3004	\n
	0x3005	לא בשימוש בגלל ה half.
var2	0x3006	0xFF
	0x3007	0xF6
	0x3008	0xFF
	0x3009	0xFF
	0x300A	לא בשימוש בגלל ה word.
	0x300B	
var3	0x300C	0x12
	0x300D	0x34
	0x300E	0x56
	0x300F	0x78

```
.DATA
var1:   .BYTE     'A', 'E', 127, -1, '\n'
var2:   .HALF     -10,  0xffff
var3:   .WORD     0x12345678
```