

# מטלת מנהה (ממ"ו) 14

הקורס : 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטרת : פרויקט גמר

משקל המטלה : 61 נקודות (חובה)

מספר השאלות : 1

מועד אחרון להגשה : 12.08.2025

סמסטר : 2025/2026

**קיימת אפשרות אחת להגשת המטלה:**  
שליחה באמצעות המטלות המקוריים באתר הבית של הקורס

**הסבר מפורט ב"נוהל הגשת מטלות מנהה"**

אחד המטרות העיקריות של הקורס "20465 - מעבדה בתכנות מערכות" היא לאפשר לסטודנטים בקורס להתנסות בכתיבת פרויקט תוכנה גדול, אשר יחקה את פעולתה של אחת מתוכניות המערכת השכיחות.

עליכם לכתוב תוכנת אסמבילר, עבור שפת אסמביל שותוגדר בהמשך. הפרויקט ייכתב בשפת C.

עליכם הגיעו את הפריטים הבאים :

1. קבצי המקור של התוכנית שתכתבם (קבצים בעלי הסיומת .c או .h).
2. קובץ הרצה (מקומפל ומקושר) עבור מערכת אוביונטו.
3. קובץ makefile. הקימפול חייב להיות עם הקומפיילר gcc והדגלים : -Wall -ansi -pedantic יש לנפות את כל החודעות שמוציאה הקומפיילר, כך שהתוכנית תתකמפל ללא כל העורות או אזהרות.
4. דוגמאות הרצה (קלט ופלט)
  - א. קובצי קלט בשפת אסמביל, קובצי הפלט שנוצרו מהפעלת האסמבילר על קבצי קלט אלה. יש להציג שימוש במגוון הפעולות וטיפוסי הנתונים של שפת האסמביל.
  - ב. קובצי קלט בשפת אסמביל המדגימים מגוון רחב של סוגים של טקסטים אסמביל וולכן לא נוצרים קבצי פלט), ותדפסי המסתן המראים את ה הודעות השגיאה שמוציאה האסמבילר. יש להגיש לפחות 3 קבצי קלט ו-3 קבצי פלט ותדפסי מסך המייצגים מצבים שונים (תקנים ושגיאות)

בשל גודל הפרויקט, עליכם לחלק את התוכנית למספר קבצי מקור, לפי מישיות. יש להקפיד שקוד המקור של התוכנית יימוד בקריטריונים של בהירות, קריאות וכתיבה נאה ומובנית.

זכור מספר היבטים חשובים של כתיבת קוד טוב :

1. הפשטה של מבני הנתונים : רצוי (כל האפשר) להפריד בין הגישה למבנה הנתונים לבין השימוש של מבני הנתונים. כך, למשל, בעת כתיבת פונקציות לטיפול בטבלה, אין זה מעניינים של משתמשים בפונקציות אלה, האם הטבלה ממומשת באמצעות מערך או באמצעות רשימה מקוشرת.
2. קריאות הקוד : יש להשתמש בשמות משמעותיים למשתנים ופונקציות. יש לעירוך את הקוד באופן מסודר: הזחות עקבות, שורות ריקות להפרדה בין קטעי קוד, ועוד'.

3. תיעוד: יש להכניס בקבצי המקור תיעוד תמציתית וברור, שיסביר את תפקידה של כל פונקציה (באמצעות העורות כתורת לכל פונקציה). כמו כן יש להסביר את תפקידם של משתנים חשובים. כמו כן, יש להכניס העורות ברמת פירוט גבוהה בכל הקוד.

הערה: תוכנית "עובדת", דהיינו תוכנית שבמצעת את כל הדריש ממנה, אינה לכשעצמה ערבבה לzion גובה. כדי לקבל zion גובה, על התוכנית לעמוד בקריטריונים של כתיבה ותיעוד ברמה גבוהה, כמפורט לעיל, אשר משקל המשותף מגע עד לכ- 40% משקל הפרויקט.

על המטלה להיות **מקורית לחלוין**: אין להיעזר בספריות חיצונית מלבד הספריות הסטנדרטיות, וכמובן לא בקוד ולא בחלקיק קוד הנמצאים בראשת, במקור חיצוני וכו'.

מומלץ לעבוד בזוגות. אין לעבוד בצוותים גדולים יותר. **פרויקט שיוגש על ידי שלשה או יותר, לא יבדק ולא יקבל zion**. חובה שסטודנטים, הבוחרים להגיש יחד את הפרויקט, יהיו **שייכים** לאותה קבוצת הנחיה. הzion יהיה לשני הסטודנטים.

מומלץ לקרוא את הגדרת הפרויקט עם ראשונה בראצ'ף, לקבלת תמורה כללית לגבי הנדרש, ורק לאחר מכן לקרוא שוב בצורה מעמיקה יותר.

### רקע כללי ומטרת הפרויקט

כידוע, קיימות שפות תוכנות רבות, ומספר גדול של תוכניות, הכתובות בשפות שונות, עשויות לדרי'ז באוטו מחשב עצמו. כיצד "מכיר" המחשב כל כך הרבה שפות? התשובה פשוטה: המחשב מכיר למעשה שפה אחת בלבד: הוראות ונתונים הכתובים בקוד בינארי. קוד זה מהו אומדן בOSH, ונראה כמו רצף של ספרות ביןarieות. יחידת העבודה המרכזית - היע"מ (CPU) - יודעת פרק את הראצ'ף הזה לקטעים קטנים בעלי משמעות: הוראות, מענים ונתונים.

למעשה, זיכרונו המחשב יכול הואוסף של סיביות, שנוהגים לארוון כמקובצות ליחידות בעלות אורך קבוע (בטים, מילימ). לא ניתן להבחין, בין שайנה מויינט, בהבדל פיסי בלבד בין אותו חלק בזיכרון שבו נמצאת תוכנית לבין שאר הזיכרונות.

יחידת העבודה המרכזית (היע"מ) יכולה לבצע מגוון פעולות פשוטות, הנקראות **הוראות מכונה**, ולשם כך היא משתמש ברגיסטרים (registers) הקיימים בתוך היע"מ, ובזיכרון המחשב. **דוגמאות:** העברת מספר מתא בזיכרון לרגיסטר בע"מ או בחזרה, הוספת 1 במספר הנמצא ברגיסטר, בדיקה האם מספר המאוחסן ברגיסטר שווה לאפס, חיבור וחיסור בין שני רגיסטרים, ועוד'.

הוראות המכונה ושילובים שלהן הן המרכיבות תוכנית כפי שהיא טעונה לזכרון בזמן ריצתה. כל תוכנית מקור (התוכנית כפי שנכתבה בידי המתכנת), תתרגם בסופו של דבר באמצעות תוכנה מיוחדת לצורה סופית זו.

היע"מ יודע לבצע קוד שנמצא בפורמט של **שפה מכונה**. זה רצף של ביטים, המהווים קידוד ביינארי של סדרת הוראות המכונה המרכיבות את התוכנית. קוד זה אינו קריא למשתמש, ולכן לא נוח לקודד (או לקובר) תוכניות ישירות בשפת מכונה. **שפה אסמבלי (assembly language)** היא שפת תוכנות מאפרשת ליציג את הוראות המכונה בצורה סימבולית קלה ונוחה יותר לשימוש. כמובן שיש צורך לתרגם את הייצוג הסימבולי לקוד בשפת מכונה, כדי שהתוכנית תוכל לרוץ במחשב. תרגום זה נעשה באמצעות כל שנקרא **אסambilר (assembler)**.

כידוע, לכל שפת תוכנות עילית יש מהדר (compiler), או מפרש (interpreter), המתרגמים תוכניות מקור לשפת מכונה. האסambilר משמש בתפקיד דומה עבור שפת אסambilי.

כל מודל של יע"מ (כלומר לכל אירוגון של מחשב) יש שפת מכונה ייעודית משלו, ובהתאם גם שפת אסambilי ייעודית משלו. לפיכך, גם האסambilר (כל התרגומים) הוא ייעודי ושונה לכל יע"מ.

תפקידו של האסambilר הוא לבנות קוד המכיל קוד מקור, מקובץ נתנו של תוכנית הכתובת בשפת אסambilי. זה השלב הראשוני בمسلسل אותו עוברת התוכנית, עד לקבלת קוד המוכן לריצה על חומרת המחשב. השלבים הבאים הם קישור (linkage) וטעינה (loading), אך בהם לא נעסק בממין זה.

המשמעות בפרויקט זה היא לכתוב אסambilר (כלומר תוכנית המתרגם לשפת מכונה), עבר שפת אסambilי שנגידר כאן במיוחד לצורך הפרויקט.

**לתשומתך:** בהסבירים הכלליים על אונן עבודת תוכנת האסטブル, תהיה מדי פעם התייחסות גם לעבודת שלבי הקישור והטעינה. התייחסויות אלה נועדו על מנת לאפשר לכם להבין את המשך תהליך העיבוד של הפלט של תוכנת האסטブル. אין לטעות: עלייכם כתוב את תוכנית האסטブル בלבד. אין לכתוב את תוכניות הקישור והטעינה!!!

### המחשב הדמיוני ושפת האסטブル

נגיד לך את שפת האסטブル ואת מודל המחשב הדמיוני, עברו פרויקט זה.  
הערה: תיאור מודל המחשב להלן הוא חלק בלבד, ככל שנחוץ לביצוע המשימות בפרויקט.

"חומרה":

המחשב בפרויקט מורכב ממעבד CPU (יע"מ - יחידת עיבוד מרכזית), רגיסטרים (אוגרים) וזיכרון RAM. חלק מהזיכרון משמש גם כמחסנית (stack).

למעבד 8 רגיסטרים כלליים,uşnames: r0, r1, r2, r3, r4, r5, r6, r7. גודלו של כל רגיסטר הוא 10 סיביות. הסיבית ה-0 היא סיבית המשמעותית ביותר (sign bit). והסיבית המשמעותית ביותר (sign bit) היא סיבית מס' 9. שמות הרגיסטרים כתובים תמיד עם אות 'r' קדימה.

כמו כן יש במעבד רגיסטר בשם PSW (program status word), המכיל מספר דגלים המאפיינים את מצב הפעולות במעבד בכל רגע נתון. ראו בהמשך, בתיאור הוראות המכונה, הסברים לגבי השימוש בדגלים אלו.

גודל הזיכרון הוא 256 תאים, בכתובות 0-255 (בבסיס עשרוני), וכל תא הוא בגודל של 10 סיביות. לתא בזיכרון נקרא גם בשם "מילה". הsilverות בכל מילה מסווגות כמו ברגיסטר.

מחשב זה עובד רק עם מספרים שלמים חיוביים ושליליים. אין תמייה במספרים ממשיים. האריתמטיקה נעשית בשיטת המשלים ל-2 (2's complement). במחשב זה יש תמייה בתווים (characters), המוצגים בקוד ascii.

מבנה הוראת מכונה:

כל הוראה מכונה במודול שלנו מורכבת מפעולה ואופרנדים. מספר האופרנדים הוא בין 0 ל-2, בהתאם לסוג הפעולה. מבחינת התפקיד של כל אופרנד, נבחין בין אופרנד מקור (source) ואופרנד יעד (destination).

כל הוראה מכונה מוקודדת במספר מיליות זיכרון רצופות, החל מ Mills **מימין** ועד **למקסימום חמש**, בהתאם לשיטת המיעון בה נתון כל אופרנד (ראו פרטיהם בהמשך).

בקובץ הפלט המכיל את קוד המכונה שבונה האסטブル, כל מילה תקודד בסיס הקסדצימלי (ראו פרטיהם לגבי קבצי פלט בהמשך).

בכל סוג הוראות המכונה, **המבנה של המילה הראשונה תמיד זהה**.  
מבנה המילה הראשונה בהוראה הוא כדלהלן:

9	8	7	6	5	4	3	2	1	0
opcode	a,b,c,d	b,c,d	c,d	opcode	source	destination	value	E,R,A	value

קיודוד כל מילה בקוד המכונה ייעשה בסיס 4 "ייחודי" המוגדר כדלקמן: ארבע ספרות הן: a,b,c,d (כאשר a שקול ל-0, b ל-1, c ל-2, d ל-3). קיודוד של מילה בגודל 10 סיביות בסיס 4 מורכב מחמש ספרות (עם ספרות a מובילות לפני הzero).

**סיביות 9-6** במליה הראשונה של הפקודה מהוות את קוד ההוראה (opcode). בשפה שלנו יש 16 קודים הוראה וهم :

הקוד בבסיס דצימלי (10)	פעולה
0	mov
1	cmp
2	add
3	sub
4	not
5	clr
6	lea
7	inc
8	dec
9	jmp
10	bne
11	red
12	prn
13	jsr
14	rts
15	stop

ההוראות נכתבות תמיד באוטיות קטנות. פרוט מושמעות ההוראות יבוא בהמשך.

#### סיביות 1-0 (A,R,E)

סיביות אלה מראות את סוג הקידוד, האם הוא מוחלט (Absolute) , חיצוני (External) או מצרייך (Relocatable)

ערך של 00 משמעו שהקידוד הוא מוחלט.

ערך של 01 משמעו שהקידוד הוא חיצוני.

ערך של 10 משמעו שהקידוד מצרייך מיקום מחדש.

סיביות אלה מתווספות רק לקידוזים של הוראות (לא של נתוניות), והן מתווספות גם לכל המילימנוטים שיש לקידוזים אלה.

**סיביות 3-2** מגדדות את מספרה של שיטת המיעון של אופרנד היעד (destination operand).

**סיביות 5-4** מגדדות את מספרה של שיטת המיעון של אופרנד המקור (source operand).

בשפה שלנו קיימות ארבע שיטות מיון, שמספרן הוא בין 0 ל-3.

השימוש בשיטות מיון מצרייך קידוד של מילוט-מידע נוספת בהוראה, לכל היותר שתי מילוטים נוספת לכל אופרנד. אם שיטת המיון של רק אחד משני האופרנדים דורשת מילוט מידע נוספת, אזו מילוט המידע הנוסף מתייחסות לאופרנד זה. אם שיטת המיון של שני האופרנדים דורשת מילוט-מידע נוספת, אזו מילוט-המידע הנוסף מתייחסות הראשון למקורו ומילוט-המידע הנוסף האחרון מתייחסות לאופרנד השני.

ארבע שיטות המיון הקיימות במכונה שלנו הן :

מספר	שיטת המיעון	תוכן מילת המידע הנוספת	אופן כתבת האופרנד	דוגמה
0	מיון מיידי	המילה הנוספת של ההוראה מכילה את האופרנד עצמה, שהוא מספר המוצג ב- 8 סיביות, אליו מתווספות זוג סיביות של שדה A,R,E	האופרנד מתחילה בטו # ולאחריו ובצמוד אליו מופיע מספר שלם בסיסי עשרוני	mov #-1,r2 בדוגמה זו האופרנד הראשון של הפקודה נטען בשיטת מיון מיידי. הפקודה כתובה את הערך -1 לתוך אוגר r2
1	מיון ישיר	המילה הנוספת של ההוראה מכילה מען של מילה בזיכרון. מען זו בזיכרו הינה האופרנד. המען מיוצג ב- 8 סיביות אליהן מתווספות זוג סיביות של שדה A,R,E	האופרנד הינו <u>תוויות</u> שהוחכרה או תוצרה בהמשך הקובץ. החזרה נעשית על ידי כתיבת <u>תוויות</u> בקובץ המקור (בחנויות).או '.data', או '.string', או '.mat', או '.extern'	dec x בדוגמה זו, תוכן המילה שבכתבות x בזיכרון ('משתנה' x) מוקtan ב-1.
2	מיון גישה למטריצה	בשיטת מיון זו משתמשים שני אוגרים ותוויות. בזמן ביצוע ההוראה, המעבד יפנה למטריצה (המצוינת ע"י התווית), לתא (בגודל מילה הנמצאת באינדקסים (שורה ועומדה) המצויים ע"י האוגרים).  בשיטת מיון זו יש 2 מילות מידע נוספת בקובץ ההוראה. המילה הנוספת הראשונה היא כתובת התחלת המטריצה (מצוין ע"י התווית). והמילה הנוספת השנייה תכיל את האינדקסים באופן הבא : ארבע הסיביות 9-6 מקודדים את מספר האוגר המכיל את אינדקס השורה, ארבע הסיביות 5-2 מקודדים את מספר האוגר המכיל את אינקס העומדה. האינדקסים מתחילהם ב-0.  לייצוג זה מתווספות זוג סיביות של שדה A,R,E.	האופרנד מורכב משם של <u>תוית</u> המציין מטריצה, ולאחריה שורה ועומדה במטריצה המצויים ע"י אוגרים בלבד, ורשומים כל אחד בסוגרים מרובעתו.	add #4, a[r2][r5] בדוגמה זו, הפקודה מוסיפה את הערך 4 לתא במטריצה שנמצאת בתוויות a. התא הוא בשורה המצוינת ע"י תוכן האוגר 2 ובעמודה המצוינת ע"י תוכן האוגר 5
3	מיון אוגר ישיר	הօפרנד הוא אוגר. אם האוגר משמש כօפרנד יעד, מילה נוספת של הפקודה תכליilar בארבע הסיביות 5-2 את מספרו של האוגר. אם האוגר משמש כօפרנד מקור, הוא יקודד במילה נוספת שתכליilar בששת הסיביות 9-6 את מספרו של האוגר.  אם בפקודה יש שני אופרנדים ושנייהם אוגרים, הם יחלקו מילה נוספת אחת משותפת. כאשר הסיביות 5-2 הן עברו אוגר היעד, והסיביות	הօפרנד הינו שם של אוגר.	mov r1,r2 בדוגמה זו, אופרנד המקור הוא האוגר r1, ואופרנד היעד הוא האוגר r2. הפקודה מעתיקה את תוכן אוגר r1 לתוך אוגר r2.  בדוגמה זו שני האופרנדים יקודדו למילה משותפת.

		9-6 הן עברו אוגר המקור. לייצוג זה מתווספות זוג סיביות של שדה A,R,E סיביות שאינן בשימוש יכלו 0.		
--	--	--	--	--

**הערה:** מותר להתייחס לתווית עוד לפני שמצהירים עליה, בתנאי שהיא אכן מוצחרת במקום כלשהו בקובץ.

#### מפורט הוראות המכונה

בתיאור הוראות המכונה השתמש במונח **PC** (קיצור של "Program Counter"). זהו רגיסטר פנימי של המעבד (לא רגיסטר כללי), שמכיל בכל רגע נתון את כתובת הוראות בה נמצאת ההוראה **הנוכחית שמתבצעת** (הכוונה תמיד לכתובת הימילה הראשונה של ההוראה).

הוראות המכונה מחלקות לשולש קבוצות, לפי מספר האופרנדים הדרוש לפעולה.

**קבוצת ההוראות הראשונות:**  
אלו הן הוראות הדורשות שני אופרנדים.

ההוראות השיקות לקבוצה זו הן: mov, cmp, add, sub, lea

הסבר הדוגמה	דוגמה	הפעולה המתבצעת	opcode	הוראה
העתק את תוכן המשטנה A (המילה שבכטובת A אל רגיסטר r1. בזיכרונו)	mov A, r1	מבצעת העתקה של תוכן אופרנד המקור (האופרנד הראשון) אל אופרנד היעד (האופרנד השני).	0	mov
אם תוכן המשטנה A זהה לתוכנו של רגיסטר r1 אז הדגל Z ("דגל האפס") ברגיסטר הסטטוס (PSW) יודלק, אחרת הדגל יאפס.	cmp A, r1	מבצעת השוואה בין שני האופרנדים. ערך אופרנד היעד (השני) מופחת מערך אופרנד המקור (הראשון), ללא שמירת תוצאות החישור. פעולה החישור מעדכנת דגל בשם Z ("דגל האפס") ברגיסטר הסטטוס (PSW).	1	cmp
רגיסטר 0 מקבל את תוצאה החיבור של תוכן המשטנה A ותוכנו הנוכחי של 0.	add A, r0	אופרנד היעד (השני) מקבל את תוצאה החיבור של אופרנד המקור (הראשון) והיעד (השני).	2	add
רגיסטר 1 מקבל את תוצאה החישור של הקבוע 3 מתוכנו הנקחי של הרגיסטר r1.	sub #3, r1	אופרנד היעד (השני) מקבל את תוצאה החישור של אופרנד המקור (הראשון) מאופרנד היעד (השני).	3	sub
המשמעות התווית HELLO מוצב לרגיסטר r1.	lea HELLO, r1	lea הוא קיצור (ראשי תיבות) של load effective address זו מצייבה את המعن בזיכרונו המוצג על ידי התווית שבאופרנד הראשון (המקורה), אל אופרנד היעד (האופרנד השני).	6	lea

### קבוצת הוראות השניה:

אלו הן הוראות הדורשות אופרנד אחד בלבד. אופן הקידוד של האופרנד הוא כמו של אופרנד היעד בפקודה עם שני אופרנדים. השדות של אופרנד המקור (סיביות 4-5) במליה הראשונה בקידוד החוראה אינם בשימוש, ולפיכך יהיה מאופסים.

ההוראות השיכנות לקבוצה זו הן: clr, not, inc, dec, jmp, bne, jsr, red, prn

הסבר הדוגמה	דוגמה	הפעולה המתבצעת	opcode	הוראה
הרגיסטר r2 מקבל את הערך 0.	clr r2	אפס תוכן האופרנד	5	clr
כל בית ברגיסטר r2 מתחף.	not r2	היפוך ערכי הסיביות באופרנד (כל סיבית שערכה 0 תהפוך ל-1 ולחיפך: 1 ל-0).	4	not
תוכן הרגיסטר r2 מוגדל ב-1.	inc r2	הגדלת תוכן האופרנד באחד.	7	inc
תוכן המשתנה Count מוקטן ב-1.	dec Count	הקטנת תוכן האופרנד באחד.	8	dec
PC ← Line	jmp Line	קפיצה (הסתעפות) בלתי מותנית אל הוראה שנמצאת במען המיצג על ידי האופרנד. כלומר, כתוצאה מביצוע ההוראה, מצביע התוכנית (PC) מקבל את ערך אופרנד היעד.	9	jmp
אם ערך הדגל Z ברגיסטר הסטטוס (PSW) הוא 0, אז PC ← Line	bne Line	בנ"ז הוא קיצור (ראשי תיבות) של: .branch if not equal (to zero) זוהי הוראה הסתעפות מותנית. אם ערכו של הדגל Z ברגיסטר הסטטוס (PSW) הינו 0, אז מצביע התוכנית (PC) מקבל את ערך הדגל Z. כזכור, הדגל Z נקבע באמצעות הוראת cmp.	10	bne
push(PC+2) PC ← address(SUBR) מצביע התוכנית קיבל את כתובת התוויות SUBR, ולבסוף, ההוראה הבאה שתבוצע תהיה במען SUBR. כתובת החזרה מהשגרה נשמרת במחסנית.	jsr SUBR	קריאה לשגרה (סברוטינה). כתובת ההוראה שאחרי הוראה jsr הנוכחיתSUBR (PC+2) נדחפת לתוך המחסנית שbiz'iroo המחשב, ומצביע התוכנית (PC) מקבל את כתובת השגרה. הערכה: חזרה מהשגרה מתבצעת באמצעות הוראת sbr, תוך שימוש בכתובת שבמחסנית.	13	jsr
קוד ה-ascii של התו הנקרא. מהקלט ייכנס לרגיסטר r1.	red r1	קריאה שלתו מהקלט הסטנדרטי (stdin) אל האופרנד.	11	red
יודפס פלט התו הנמצא בSEGMENT, אל הנמצא ברגיסטר r1	prn r1	הדפסת התו הנמצא בSEGMENT, אל הפלט הסטנדרטי (stdout).	12	prn

### **קבוצת הוראות השלישייה:**

אלו הן הוראות ללא אופרנדים. קידוד ההוראה מרכיב ממילה אחת בלבד. השדות של אופרנד המקור ושל אופרנד היעד (סיביות 5-2) במילה הראשונה של קידוד ההוראה אינם בשימוש, ולפיכך יהיו מאופסים.

הhorאות השויות לקבוצה זו הן : rts, stop .

הסבר הדוגמה	דוגמיה	הפעולה המתבצעת	opcode	הוראה
PC $\leftarrow$ pop()  ההוראה הבאה שתתבצע תהיה זו שאחרי הוראת jsr שקרה לשגרה.	rts	מתבצעת חזרה משיגרה. הערך שבראש המחסנית של המחשב מצוי מן המחסנית, ומוכנס למצוין התוכנית (PC). הערה : ערך זה נכנס למחסנית בקריאה לשגרה ע"י הוראת jsr	14	rts
התוכנית עצרת מיידית.	stop	עצירת ריצת התוכנית.	15	stop

### מבנה שפת האסמבלי :

תכנית בשפת אסמבלי בנויה **ממקראים וMESSAGES** (statements).

### מקראים :

מקראים הם קטעי קוד הכלולים בתוכם משפטים. בתוכנית ניתן להגדיר מקאו ולהשתמש בו במקומות שונים בתוכנית. השימוש במקאו ממוקם מסוים בתוכנית יגרום לפרישת המקאו לאותו מקום.

הגדרת מקאו נעשית באופן הבא : (בדוגמה שם המקאו הוא a\_mc)

```
macro a_mc
    inc r2
    mov A,r1
endmacro
```

שימוש במקאו הוא פשוט אזכור שמו.  
למשל, אם בתוכנית במקום כלשהו כתוב :

```
a_mc
```

```
a_mc
```

בדוגמה זו, השתמשנו פעמיים במקאו a\_mc, התוכנית לאחר פרישת המקאו תיראה כך :

```
inc r2
mov A,r1

inc r2
mov A,r1
```

## התוכנית לאחר פרישת המאקרו היא התוכנית שהאסטמבלר אמור לתרגם.

### הנחות והניחות לגבי מאקרו:

- אין במערכת הגדרות מאקרו מוקנות (אין צורך לבדוק זאת).
- שם של הוראה או הנחה לא יכול להיות שם של מאקרו (יש לבדוק זאת).
- ניתן להניח שלכל שורת מאקרו בקוד המקור קיימת סגירה עם שורת microend (אין צורך לבדוק זאת).
- הגדרת מאקרו תהיה תמיד לפני הקריאה למאקרו (אין צורך לבדוק זאת).
- נדרש שהקדם-אסטמבלר ייצור קובץ עם הקוד המורחב הכלול פרישה של המאקרו (הרחבת של קובץ המקור המתואר בהמשך). "קובץ המקור המורחב" הוא "קובץ מקור" לאחר פרישת המאקרו, לעומת "קובץ מקור ראשוני" שהוא קובץ הקלט למערכת, כולל הגדרת המאקרוים.

### לסיכום, במאקרו יש לבדוק:

- (1) שם המאקרו תקין (אינו שם הוראה וכדומה)
  - (2) בשורת ההגדרה ובשורת הסיום אין תווים נוספים
- אם נמצאה שגיאה בשלב פרישת המאקרו - אי אפשר לעبور לשלבים הבאים:  
יש לעזoor להודיע על השגיאות ולחזור לקובץ המקור הבא (אם קיים).
- הערה: **שגיאות בגוף המאקרו** (אם יש) מגלים בשלבים הבאים.

### משפטים:

קובץ מקור בשפט אסטמבלי מורכב משורות המכילות משפטיים של השפה, כאשר כל משפט מופיע בשורה נפרדת. כלומר, הפרדה בין משפט בקובץ המקור הינה באמצעות התו 'ח' (שורה חדשה).

אורכה של שורה בקובץ המקור הוא 80 תווים לכל היותר (לא כולל התו 'ח'). אם השורה ארוכה מ- 80 תווים, יש לדוח על שגיאה.

יש ארבעה סוגי משפטיים (שורות בקובץ המקור) בשפט אסטמבלי, והם :

סוג המשפט	הסביר כלל
משפט ריק	זהו שורה המכילה אך ורק תווים לבנים (whitespace), ככלומר רק את התווים ' ' ו- 'ז' (רווחים וטאבים). יתכן ו בשורה אין אף תו (למעט התו 'ח'), ככלומר השורה ריקה.
משפט הערת	זהו שורה בה התו הראשון הינו ',' (נקודה פסיק). על האסטטטול להתעלם לחולטן משורה זו.
משפט הנחה	זהו משפט המנחה את האסטטטול מה עליו לעשות כשהוא פועל על תוכנית המקור. יש מספר סוגים של משפטי הנחה. משפט הנחה עשוי לגורם להקצת זיכרון ואתחול משתנים של התכנית, אך הוא אינו מייציר קידוד של הוראות מכונה המיועדות לביצוע בעת ריצת התכנית.
משפט הוראה	זהו משפט המייציר קידוד של הוראות מכונה לביצוע בעת ריצת התכנית. המשפט מורכב ממשם של הוראה שעל המעבד לבצע, ותיאור האופרנדים של הוראה.

cutet נפרט יותר לגבי סוגי המשפטים השונים.

### משפט הנחה:

משפט הנחה הוא בעל המבנה הבא :

בתחילת המשפט יכולה להופיע הגדרה של **תווית** (label). לתווית יש תחביר חוקי, שיתוואר בהמשך. התווית היא אופציונלית.

לאחר מכן מופיע שם הנחה. לאחר שם הנחה יופיעו פרמטרים (מספר הפרמטרים בהתאם להנחה). שם של הנחה מתחליבתו ',' (נקודה) ולאחריו תווים באותיות קטנות (lower case) בלבד.

**יש לשים לב: למילים בקוד המכונה הנוצרות משפט הוכח לא מצורף השדה E,R,A, והקידוד מלא את כל הסיבות של המילה.**

יש ארבעה סוגים של משפטי הוכח, וهم :

### 1. ההנחה 'data'.

הפרמטרים של ההנחה 'data', הם מספרים שלמים חוקיים (אחד או יותר) המופרדים על ידי התו ',' (פסיק). לדוגמה :

.data 7, -57, +17, 9

יש לשים לב שהפסיקאים אינם חייבים להיות צמודים למספרים. בין מספר לפסיק ובין פסיק למספר יכולים להופיע רווחים וטאים בכל כמות (או בכלל לא), אולם הפסיק חייב להופיע בין המספרים. כמו כן, אסור שיוופיע יותר מפסיק אחד בין שני מספרים, וגם לא פסיק אחרי המספר האחרון או לפני המספר הראשון.

המשפט 'data' מנהה את האSEMBLER להקצות מקומות בתמונה הנתונים (data image), אשר בו יוחסנו הערךם של הפרמטרים, ולאחר מכן מונח הנתונים, בהתאם למספר הערךם. אם בהנחה מוגדרת תווית, אז תווית זו מקבלת את ערך מונח הנתונים (לפני הקידום), ומוכנסת אל data. טבלת הסמלים. דבר זה מאפשר להתייחס אל מקום מסוים בתמונה הנתונים דרך שם התווית (למעשה, זהה דרך להגדיר שם של משתנה).

כלומר אם נכתב :

XYZ: .data 7, -57, +17, 9

אז יוקצו בתמונה הנתונים ארבע מילים רצופות שיכילו את המספרים שמופיעים בהנחה. התווית XYZ מזוהה עם כתובות המילה הראשונה.

אם נכתב בתכנית את ההוראה :

mov XYZ, r1

אז בזמן ריצת התכנית יוכנס לרגיסטר r1 ערך 7.

ואילו ההוראה :

lea XYZ, r1

תכניס לרגיסטר r1 את ערך התווית XYZ (כלומר הכתובת בזיכרונו בה מאוחSEN הערך 7).

### 2. ההנחה 'string'.

להנחה 'string', פרמטר אחד, שהוא מחרוות חוקית. תוויה מחרוות מקודדים לפי ערכי ASCII המתאימים, ומוכנסים אל תמונה הנתונים לפי סדרם, כלתו במלחה נפרדת. בסוף המחרוזת יתוסף התן '0' (הערך המספרי 0), המסמן את סוף המחרוזת. מונח הנתונים של האSEMBLER יקודם בהתאם לאורך המחרוזת (בתוכסת מקום אחד עברו התו המסיים). אם בשורת ההנחה מוגדרת תווית, אז תווית זו מקבלת את ערך מונח הנתונים (לפני הקידום) ומוכנסת אל טבלת הסמלים, בדומה למה שנעשה עבור 'data'. (כלומר ערך התווית יהיה הכתובת בזיכרונו שבה מתחלפת המחרוזת).

לדוגמה, ההנחה :

STR: .string "abcdef"

מקצת בתמונה הנתונים רצף של 7 מילים, ומאותחלת את המילים לקודי ascii של התווים לפי הסדר במחירות, ולאחריהם הערך 0 לסימון סוף מחירות. התוויות STR מזוהה עם כתובות התחלות המחרוזת.

### 3. '.mat'

משפט הינה זה מקצת מטריצה. למשפט הינה '.mat'. המבנה הבא:

MAT8: .mat [2][3]  
MAT5: .mat [2][2] 4,-5,7,9

בדוגמא הראשונה מקצים מטריצה בשם MAT8 מוגדרת 2 שורות ו-3 עמודות והיא אינה מאותחלת בערכים (אבל כן צורכת מקום בתמונה הנתונים).

בדוגמא השנייה מקצים מטריצה בשם MAT5 בגודל 2 שורות ו-2 עמודות, המאותחלת לערכים המפורטים. הערכים לאთחולו ורק שמאל לימין לפי סדר השורות.  
מטריצה תכיל רק מספרים שלמים.

### 4. '.entry'

להנעה '.entry' פרמטר והוא שם של תוויות המוגדרת בקובץ המקור הנוכחי (כלומר תווית שמקבלת את ערכה בקובץ זה). מטרת ההנעה entry. היא לאפיין את התוויות הזו באופן שיאפשר לקוד אסטטטי נמוך בקבצי מקור אחרים להשתמש בה (כאופרנד של הוראה).

לדוגמה, השורות:

```
.entry    HELLO  
HELLO: add    #1, r1  
.....
```

מודיעות לאסטטטי שאפשר להתייחס בקובץ אחר לתוית O HELLO המוגדרת בקובץ הנוכחי.

לשומות לב: תווית המוגדרת בתחילת שורת entry. הינה חסרת משמעות והאסטטטי מועלם מתווית זו (אפשר שהאסטטטי יוציא הודעה אזהרה).

### 5. '.extern'

להנעה '.extern' פרמטר והוא שם של תוויות שאינה מוגדרת בקובץ המקור הנוכחי. מטרת ההוראה היא להודיע לאסטטטי כי התוויות מוגדרת בקובץ מקור אחר, וכי קוד האסטטטי בקובץ הנוכחי עושה בתווית שימוש.

נשים לב כי הנעה זו תואמת להנעה 'entry'. המופיעה בקובץ בו מוגדרת התוויות. בשלב הקישור תבוצע התאמה בין ערך התוית, כפי שנקבע בקוד המכונה של הקובץ שהגדיר את התוית, לבין קידוד ההוראות המשמשות בתוית בקבצים אחרים (שלב הקישור אינו רלוונטי למynn' זה).

לדוגמה, המשפט ההנעה '.extern' התואם למשפט ההנעה 'entry' מהדוגמה הקודמת יהיה:

```
.extern    HELLO
```

**הערה:** לא ניתן להגדיר באותו הקובץ את אותה התווית גם `extern` וגם `c-extern` (בדוגמאות לעיל, התווית `HELLO`). במקרה כזה יש להודיע על שגיאת.

**لتשומת לב:** תווית המוגדרת בתחילת שורת `extern`. הינה חסרת משמעות והאסמבלר **מתעלם** מהתווית זו (אפשר שהאסמבלר יוציא הודעה אזהרה).

#### משפט הוראה:

משפט הוראה מורכב מחולקים הבאים:

1. תווית אופציאונלית.
2. שם הפעולה.
3. אופרנדים, בהתאם לסוג הפעולה (בין 0 ל-2 אופרנדים).

אם מוגדרת תווית בשורת ההוראה, אז היא תוכנס אל טבלת הסמלים. ערך התווית יהיה מען המילה הראשונה של ההוראה בתוך תמונה הקוד שבונה האסמבלר.

שם הפעולה תמיד בתוויות קטנות (lower case), והוא אחת מ- 16 הפעולות שפורטו לעיל.  
לאחר שם הפעולה יופיעו האופרנדים, בהתאם לסוג הפעולה. יש להפריד בין שם הפעולה לבין האופרנד הראשון באמצעות רווחים ו/או ט-abs (אחד או יותר).

כאשר יש שני אופרנדים, האופרנדים מופרדים זה מהו בפסיק. בדומה להנחיה '`data`', **לא חייבת להיות הצמדה של האופרנדים לפסיק**. כל כמה של רוחניים ו/או ט-abs משני צידי הפסיק היא חוקית.

למשפט הוראה עם שני אופרנדים המבנה הבא:

label: opcode source-operand, target-operand

לדוגמה:

HELLO: add r7, B

למשפט הוראה עם אופרנד אחד המבנה הבא:

label: opcode target-operand

לדוגמה:

HELLO: bne XYZ

למשפט הוראה ללא אופרנדים המבנה הבא:

label: opcode

לדוגמה:

END: stop

#### אפיון השדות במשפטים של שפת האסמל

**תווית:**

בתיאור שיטות המידע לעלה הסברנו כי **תווית** היא ייצוג סימבולי של כתובות בזיכרון. נרჩיב כאן את ההסביר:  
תווית היא למעשה סמל שМОוגדר בתחילת שפת הוראה, או בתחילת הנחיתת `data`. או `string`.  
תווית חוקית מתחילה באות אלפביתית (גדולה או קטנה), ולאחריה סדרה של אותיות אלפביתיות (גדולות או קטנות) ו/או ספרות. האורך המקסימלי של תווית הוא 30 תווים.

הגדרה של תווית מסתויימת בתו : (נקודתיים).תו זה אינו מהוות חלק מהתווית, אלא רק סימן המציין את סוף ההגדרה. התו ',' חייב להיות צמוד לתווית (לא רווחים).

אסור שאותה תווית תוגדר יותר מפעם אחת (כਮון בשורות שונות).אותיות קטנות וגדולות נחשות שונות זו מזו.

לדוגמא, התוויות המוגדרות להלן הן תוויות חוקיות.

hEllo:

x:

He78902:

**لتשומת לב :** מילים שמורות של שפת האסמבלי (כלומר שם של פעולה או הינה, או שם של רגיסטר) אינן יכולות לשמש גם כשם של תווית. כמו כן, אסור שאותו סמל ישמש הן כתווית והן שם של מקאו (יש לבדוק זאת).

התווית מקבלת את ערכה בהתאם להקשר בו היא מוגדרת. תווית המוגדרת בהנחיות `.data`, `.string`, `.mat`. קיבלת ערך מונה הנתונים (data counter) הנוכחי, בעוד שתווית המוגדרת בשורת הוראה קיבלת ערך מונה ההוראות (instruction counter) הנוכחי.

**لتשומת לב :** מוטר במשפט הוראה להשתמש באופrnd שהוא סמל שאינו מוגדר כתווית בקובץ הנוכחי, כל עוד הסמל מאופיין כחיצוני (באמצעות הנחיה `extern`. כלשיי בקובץ הנוכחי).

#### מספר :

מספר חוקי מתחילה בסימן אופצionalי: '-' או '+', ולאחריו סדרה כלשהי של ספרות בסיס עשרוני. לדוגמה: -5, +123, 76 הם מספרים חוקיים. אין תמיכה בשפת האסמבלי שלנו ביצוג בסיס אחר מאשר עשרוני, ואין תמיכה במספרים שאינם שלמים.

#### מחירות :

מחירות חוקית היא סדרת תוו ascii נראים (שניתנים להדפסה), המוקפים במרקאות כפولات (המרקאות אינן נחבות חלק מהמחירות). דוגמה למחירות חוקית: "hello world".

#### "**Sימון המילים בקוד המכונה באמצעות המאפיין "A,R,E"**

בכל מילה בקוד המכונה של הוראה (לא של נתונים), האסמבלי מכין מידע עבור תהליך הקישור והטעינה. זה השדה A.R,E. המייד ישתמש לתיקונים בקוד בכל פעם שייטענו לזכרו לצורך הרצה. האסמבלי בונה מלכתחילה קוד שמיועד לטיענה החל מכתובת התחלה. התיקונים אפשרו לטעון את הקוד בכל פעם למקום אחר, בלי צורך לחזור על תהליך האסמבלי.

שלוש הסיבות בשדה A.R,E יכilo ערכאים בינאריים כפי שהסביר בתיאור שיטות המיעון. המשמעות של כל ערך מפורטת להלן.

האות 'A' (קייזר של absolute) בא לציין שתוקן המילה אינו תלוי במקומות בזיכרון בו ייטען בפועל קוד המכונה של התכנית בעת ביצועה (למשל מילה המכילה אופrnd מיידי).

האות 'R' (קייזר של relocatable) בא לציין שתוקן המילה תלוי במקומות בזיכרון בו ייטען בפועל קוד המכונה של התכנית בעת ביצועה (למשל מילה המכילה כתובת של תווית המוגדרת בקובץ המקור).

האות 'E' (קייזר של external) בא לציין שתוקן המילה תלוי בערכו של סמל חיצוני (external) (למשל מילה המכילה כתובת של תווית חיצונית, ככלומר תווית שאינה מוגדרת בקובץ המקור).

כאשר האסמבלי מקבל קלק תוכנית בשפת אסמבלי, עליו לטפל תחילת בפרישת המאקרוואים, ורק לאחר מכן לעבור על התוכנית אליה נפרשו המאקרוואים. ככלומר, פרישת המאקרוואים תעשה בשלב "קדם אסמבלי", לפני שלב האסמבלי (המtoaר בהמשך).

אם התוכנית אינה מכילה מאקרו, תוכנית הפרישה תהיה זהה לתוכנית המקורי.

דוגמה לשלב קדם אסמבולר. האסמבולר מקבל את התוכנית הבאה בשפת אסמבולר:

```
MAIN:    mov    M1[r2][r7],LENGTH
          add    r2,STR
LOOP:    jmp    END
          prn    #-5
          mero   a_mc
          mov    M1[r3][r3],r3
          bne    LOOP
          meroend
          sub    r1, r4
          inc    K

          a_mc
END:     stop
STR:     .string "abcdef"
LENGTH:  .data  6,-9,15
K:       .data  22
M1:      .mat [2][2] 1,2,3,4
```

תחילת האסמבולר עובר על התוכנית ופורש את כל המאקרוואים הקיימים בה. רק אם מתקיים זה מסתויים בהצלחה, ניתן לעבור לשלב הבא. אחרת, יש להציג את השגיאות ולא לייצר קבצים. בדוגמה זו, התוכנית לאחר פרישת המאקרו תיראה כך:

```
MAIN:    mov    M1[r2][r7],LENGTH
          add    r2,STR
LOOP:    jmp    END
          prn    #-5
          sub    r1, r4
          inc    K

          mov    M1[r3][r3],r3
          bne    LOOP
END:     stop
STR:     .string "abcdef"
LENGTH:  .data  6,-9,15
K:       .data  22
M1:      .mat [2][2] 1,2,3,4
```

קוד התוכנית, לאחר הפרישה, ישמר בקובץ חדש, כפי שIOSBVR בהמשך.

## אלגוריתם שלדי של קדם האסמבילר

נציג להלן אלגוריתם שלדי לתחילה קדם האסמבילר. لتשומת לב: אין חובה להשתמש דווקא באלגוריתם זה :

- .1 קרא את השורה הבאה מקובץ המקור. אם נגמר הקובץ, עבור ל- 9 (סיום).
- .2 האם השדה הראשון הוא שם מאקרו המופיע בטלטת המאקרו (כגון a\_mc)? אם כן, החלף את שם המאקרו והעתק במקומו את כל השורות המתאימות מהטבלה לקובץ, חזרו ל- 1. אחרת, המשך.
- .3 האם השדה הראשון הוא "mcr" (התחלת הגדרת מאקרו)? אם לא, עבור ל- 6.
- .4 הדלק דגל "יש mcr".
- .5 (קייםת הגדרת מאקרו) הכנס לטבלת שורות מאקרו את שם המאקרו (לדוגמא a\_mc).
- .6 קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עבור ל- 9 (סיום).
- .7 האם דגל "יש mcr" דולק ולא זוהה תווית microend? הכנס את השורה לטבלת המאקרו ומחק את השורה הניל' מהקובץ. אחרת (לא מקacro) חזרו ל- 1.
- .8 כבה דגל "יש mcr". חזרו ל- 1. (סיום שמירת הגדרת מאקרו).
- .9 סיום : שמירת קובץ מאקרו פרוש.

## אסמבילר עם שני מעברים

במעבר הראשון של האסמבילר, יש לזהות את הסמלים (תוויות) המופיעים בתוכנית, ולתת לכל סמל ערך מסוימי שהוא המعنוי בזיכרון שהSAMPLE מייצג. במעבר השני, באמצעות ערכי הסמלים, וכן קוד-הפעולה ומספריו הרגיסטריים, בונים את קוד המכונה.

עליו להחליף את שמות הפעולות בקוד הבינארי השקול להם במודל המחשב שהגדנו. כמו כן, על האסמבילר להחליף את כל הסמלים (למשל LOOP, MAIN ) במשמעותם של המקומות בזיכרון שם נמצאים כל נתון או הוראה בהתקאה.

נניח שقطع הקוד לעיל (הוראות ונתונים) ייטע בזיכרון החל ממען 100 (בסיס 10). במקרה זה נקבל את ה"תרגום" הבא :

لتשומת לב : המקרים המופיעים בקידוד הבינרי הם רק לצורך הדגשת ההפרדה בין השינויים בקידוד ונוועדו לשם המוחשה בלבד.

Label	Decimal Address	Base 4 Address	Instruction	Operands	Binary machine code
MAIN:	0100	1210	mov	M1[r2][r7],LENGTH	0000-10-01-00
	0101	1211		M1	10000101-10
	0102	1212		קידוד אוגרי האינדקסים במטריצה	0010-0111-00
	0103	1213		כתובת של LENGTH	10000001-10
	0104	1220	add	r2, STR	0010-11-01-00
	0105	1221		קידוד מספר האוגר	0010-0000-00
	0106	1222		כתובת של STR	01111010-10
LOOP:	0107	1223	jmp	END	1001-00-01-00
	0108	1230		כתובת של END	01111001-10
	0109	1231	prn	#-5	1100-00-00-00

	<b>0110</b>	1232			-5	המספר	<b>11111011-00</b>
	<b>0111</b>	1233	<b>sub</b>	r1,r4		<b>0011-11-11-00</b>	
	<b>0112</b>	1300			קידודי מספרי האוגרים	<b>0001-0100-00</b>	
	<b>0113</b>	1301	<b>inc</b>	K	כתובת של K	<b>0111-00-01-00</b>	
	<b>0114</b>	1302				<b>10000100-10</b>	
	<b>0115</b>	1303	<b>mov</b>	M1[r3][r3],r3	כתובת של M1	<b>0000-10-11-00</b>	
	<b>0116</b>	1310			קידוד אוגרי האינדקסים במטריצה	<b>10000101-10</b>	
	<b>0117</b>	1311			קידוד מספר האוגר של היעד	<b>0011-0011-00</b>	
	<b>0118</b>	1312				<b>0000-0011-00</b>	
	<b>0119</b>	1313	<b>bne</b>	LOOP	כתובת של LOOP	<b>1010-00-01-00</b>	
	<b>0120</b>	1320				<b>01101011-10</b>	
<i>END:</i>	<b>0121</b>	1321	<b>stop</b>			<b>1111-00-00-00</b>	
<i>STR:</i>	<b>0122</b>	1322	<b>.string</b>	“abcdef”		<b>0001100001</b>	
	<b>0123</b>	1323				<b>0001100010</b>	
	<b>0124</b>	1330				<b>0001100011</b>	
	<b>0125</b>	1331				<b>0001100100</b>	
	<b>0126</b>	1332				<b>0001100101</b>	
	<b>0127</b>	1333				<b>0001100110</b>	
	<b>0128</b>	2000				<b>0000000000</b>	
<i>LENGTH:</i>	<b>0129</b>	2001	<b>.data</b>	6,-9,15		<b>0000000110</b>	
	<b>0130</b>	2002				<b>1111110111</b>	
	<b>0131</b>	2003				<b>0000001111</b>	
<i>K:</i>	<b>0132</b>	2010	<b>.data</b>	22		<b>0000010110</b>	
<i>M1</i>	<b>0133</b>	2011	<b>.mat</b>	[2][2] 1,2,3,4		<b>0000000001</b>	
	<b>0134</b>	2012				<b>0000000010</b>	
	<b>0135</b>	2013				<b>0000000011</b>	
	<b>0136</b>	2020				<b>0000000100</b>	

האסמבלר מחזיק טבלה שבה רשומים כל שמות הפעולה של ההוראות והקודים הבינאריים המתאים להם, וכן שמות הפעולות ניתנים להמרה לבינארי בקבוצות. כאשר נקרא שם פעולה, אפשר פשוט לעיין בטבלה ולמצוא את הקוד הבינארי.

כדי לבצע המרה לבינארי של אופרנדים שכותבים בשיטות מייען המשמשות בסמלים (תוויות), יש ליצור לבנות טבלה המכילה את ערכיו כל הסמלים. אולם בהבדל מהקודים של הפעולות, הידועים מראש, הרי המענים בזיכרונו עבור הסמלים ששימוש התוכנית אינם ידועים, עד אשר תוכנית המקור נסקרה כולה ונתגלו כל הגדרות הסמלים.

למשל, במקרה לעיל, האסמבלר יוכל לדעת שהסמל END משוויך למען 121 (עשרוני), אלא רק לאחר שנקרוו כל שורות התוכנית.

לכן מפרידים את הטיפול של האסמבלר בסמלים לשני שלבים. בשלב הראשון בונים טבלה של כל הסמלים, עם הערכים המספריים המשווים להם, ובשלב השני מחליפים את כל הסמלים, המופיעים באופרנדים של הוראות התוכנית, בערכיהם המספריים. הביצוע של שני שלבים אלה כרוך בשתי סריקות (הנקראות "מעברים") של קובץ המקור.

במעבר הראשון נבנית טבלת סמלים בזיכרון, ובה לכל סמל שבתוכנית המקור משוויך ערך מספרי, שהוא מען בזיכרון. בדוגמה לעיל, טבלת הסמלים לאחר מעבר ראשון היא:

סמל	ערך (בסיס עשרוני)
MAIN	100
LOOP	107
END	121
STR	122
LENGTH	129
K	132
M1	133

במעבר השני נעשית ההמרה של קוד המקור לקוד מוכנה. בתחילת המעבר שניים מרכזיים הערכים של הסמלים להיות כבר ידועים.

לשומות לב: תפקיד האסטבלר, על שני המעברים שלו, לתרגם קובץ מקור לקוד בשפת מוכנה. בגמר פעולה האסטבלר, התכנית טרם מוכנה לטעינה לזכרו לצורך ביצוע. קוד המוכנה חייב לעבור שלבי הקישור/טעינה, ורק לאחר מכן לשלב הביצוע (שלבים אלה **אין** חלק מהמ מב'ו).

### המעבר הראשון

במעבר הראשון נדרשים כללים כדי לקבוע איזה מען ישוק לכל סמל. העיקרונות הבסיסיים הוא לספור את המיקומות בזיכרון, אותן תפוזות ההוראות. אם כל הוראה תיעטו בזיכרון למקום העוקב להוראה הקודמת, תציין ספרה כזאת את מען ההוראה הבאה. הספרה נועשת על ידי האסטבלר ומוחזקת במונה ההוראות (IC). ערכו ההתחלתי של IC הוא 100 (עשורי), ולכן המוכנה של ההוראה הראשונה נבנה כך שייעטן לזכרו החל מען 100. ה-IC מתעדכן בכל שורת הוראה המקצת מקום בזיכרון. לאחר שהאסטבלר קובע מהו אורך ההוראה, ה-IC מוגדל במספר התאים (מילימטרים) הנתפסים על ידי ההוראה, וכך הוא מצביע על התא הפוני הבא.

כאמור, כדי לקודד את ההוראות בשפת מוכנה, מחייב האסטבלר טבלה, שיש בה קוד מתאים לכל שם פעולה. בזמן התרגום מחליף האסטבלר כל שם פעולה בקוד שלה, וכן כל אופרנד מוחלף בקידוד מתאים, אך פעולה הינה זיהוי בלבד. ההוראות משתמשות בשיטות מייען מגוונות לאופרנדיים. אותה פעולה יכולה לקבל שימושיות שונות, בכל אחת משיטות המייען, וכן יתאפשרו לה קידודים שונים לפי שיטות המייען. לדוגמה, פעולה ההזזה **sum** יכולה להתיחס להעתקת תוכן תא זיכרון לריגיסטר, או להעתקת תוכן לריגיסטר אחר, וכן הלאה. לכל אפשרות כזו של **sum** עשוי להתאים קידוד שונה.

על האסטבלר לסרוק את שורת ההוראה בשלמותה, ולהחיליט לגבי הקידוד לפי האופרנדיים. בדרך כלל מתחלק הקידוד לשדה של שם הפעולה, ושדות נוספים המכילים מידע לגבי שיטות המייען. כל השדות ביחד דורשים מילה אחת או יותר בקוד המוכנה.

כאשר נתקל האסטבלר בתווית המופיעה בתחילת ההוראה, הוא יודע שלפני הגדרה של תווית, והוא הוא משייך לה מען – תוכנו הנוכחי של ה-IC. כך מקבלות כל התוויות את מעניין בעת ההגדירה. תוויות אלה מוכנסות לטבלת הסמלים, המכילה בנוסף לשם התווית גם את המען ומאפיינים נוספים. כאשר תהיה התייחסות לתווית באופרנד של הוראה כלשהי, יוכל האסטבלר לשולח את המען המתאים מטבלת הסמלים.

הוראה יכולה להתיחס גם לסמל שטרם הוגדר עד כה בתכנית, אלא יוגדר רק בהמשך התכנית. להלן לדוגמה, הוראות הסתעפות מען שמוגדר על ידי התווית A שמשמעותו רק בהמשך הקוד :

bne      A  
.....  
.  
.  
.  
A:        .....

כאשר מגיע האסטבלר לשורת ההסתעפות (A bne), הוא טרם נתקל בהגדרת התווית A וכמוון לא יודע את המען המשויך לתווית. לכן האסטבלר לא יכול לבנות את הקידוד הבינאי של האופרנד A. נראה בהמשך כיצד נפתרת בעיה זו.

בכל מקרה, תמיד אפשר לבנות במעבר הראשון את הקידוד הבינאי המלא של המילה הראשונה של כל הוראה, את הקידוד הבינאי של מילת-המידע הנוספת של אופרנד מיידי, או ריגיסטר, וכן את הקידוד הבינאי של כל הנתונים (המקבלים מההנחיות .mat, .string, .data).

## המעבר השני

ראינו שבמעבר הראשון, האסטובלר אין יכול לבנות את קוד המcona של אופרנדים המשתמשים בסמלים שעדין לא הוגדרו. רק לאחר שאסטובלר עבר על כל התכנית, כך שכל הסמלים נכנסו כבר לטבלת הסמלים, יוכל האסטובלר להשלים את קוד המcona של כל האופרנדים.

לשם כך מבצע האסטובלר מעבר נוסף (מעבר שני) על כל קובץ המקור, ומעדכו את קוד המcona של האופרנדים המשתמשים בסמלים, באמצעות ערכי הטבלת הסמלים מטבלת הסמלים. בסוף המעבר השני, תהיה התוכנית מתורגמת בשלמותה לקוד מכונה.

### הפרזה הוראות ונתונים

בתכנית מבחןים בשני סוגים של תוכן: הוראות ונתונים. יש לארגן את קוד המcona כך שתיהיה הפרדה בין הנתונים וההוראות. הפרזה הוראות והנתונים לקטומים שונים בזיכרון היא שיטה עדיפה על פני הצמדה של הגדרות הנתונים להוראות המשמשות בהן.

אחרת הסכנות הטമונות באירוע ההוראות מהנתונים היא, שימושים עלול המעבד, בעקבות שגיאה לוגית בתכנית, לנסות "לבצע" את הנתונים כאילו היו הוראות חוקיות. למשל, שגיאה שיכולה לגרום לתגובה לא נכונה. התכנית כמובן לא תעבור נכוון, אך לרוב הנזק הוא יותר חמור, כי נוצרת חריגת חומרה ברגע שהמעבד מבצע פעולה שאינה חוקית.

האסטובלר שלנו **חייב** להפריד, בקוד המcona שהוא מייצר, בין קטע הנתונים לבין הוראות. **כלומר בקובץ הפלט (בקוד המcona) תהיה הפרזה של הוראות ונתונים לשני קטעים נפרדים, ואילו בקובץ הקטל אין צורך שתהיה הפרזה בזו.** בהמשך מתואר אלגוריתם של האסטובלר, ובו פרטיים כיצד לבצע את הפרזה.

### גילוי שגיאות בתכנית המקור

הנחת המטלה היא **שאין שגיאות בהגדירות המאקרו**, ולכן שלב קדם האסטובלר אין מカリ שלב גילוי שגיאות. אין גם צורך לבדוק שגיאות בפתרונות / סגירת המאקרו (למשל אם המאקרו לא מסתתיים – ניתן להניח שהנהיל תקין). לעומת זאת, האסטובלר אמר לגלות ולדווח על **שגיאות בתחריר של תוכנית המקור**, כגון פעולה שאינה קיימת, מספר אופרנדים שגוי, סוג אופרנד שאינו מתאים לפעולה, שם רגיסטר לא קיים, ועוד שגיאות אחרות. כמו כן מודיא האסטובלר שככל סמל מוגדר פעמי אחד בדיק.

מכאן, שככל שגיאה המתגלגה על ידי האסטובלר נגרמת (בדרך כלל) על ידי שורת קלט מסוימת. לדוגמה, אם מופיעים שני אופרנדים בהוראה שאמור להיות בה רק אופרנד יחיד, האסטובלר ייתן הודעה שגיאה בנוסח "יוטר מדי אופרנדים".

הערה: אם יש שגיאה בקוד האסטובלרי בווקס מאקרו, הרוי שגיאה זו יכולה להופיע ולהתגלות שוב ושוב, בכל מקום בו נפרש המאקרו. נשים לב שכאשר האסטובלר בודק שגיאות, כבר לא ניתן לזיהות שזה קוד שנפרש ממאקרו, כך שלא ניתן לחסוך גילוי שגיאה כפоляים.

האסטובלר ידפיס את ה הודעות השגיאה אל הפלט הסטנדרטי `stdout`. בכל הודעה שגיאה יש לציין גם את מספר השורה בקובץ המקור בה זוחתה השגיאה (מנין השורות בקובץ מתחילה ב-1).

**لتשומת לב:** האסטובלר **אין עוצר** את פעולתו אחרי שנמצאה השגיאה הראשונה, אלא ממשיך לעبور על הקטל כדי לגלוות שגיאות נוספות, ככל שישן. כמובן שאין כל טעם לייצר את קבצי הפלט אם נתגלו שגיאות (ممילא אי אפשר להשלים את קוד המcona).

הטבלה הבאה מפרטת מהן שיטות המיעון החוקיות, עבור אופרנד המקור ואופרנד היעד של ההוראות השונות הקיימות בשפה הנתונה:

שיטות מייען חוקיות עבור אופrnd היעד	שיטות מייען חוקיות עבור אופrnd המקור	שם ההוראה	Opcode
1,2,3	0,1,2,3	mov	0
0,1,2,3	0,1,2,3	cmp	1
1,2,3	0,1,2,3	add	2
1,2,3	0,1,2,3	sub	3
1,2,3	1,2	lea	4
1,2,3	אין אופrnd מקור	clr	5
1,2,3	אין אופrnd מקור	not	6
1,2,3	אין אופrnd מקור	inc	7
1,2,3	אין אופrnd מקור	dec	8
1,2,3	אין אופrnd מקור	jmp	9
1,2,3	אין אופrnd מקור	bne	10
1,2,3	אין אופrnd מקור	jsr	11
1,2,3	אין אופrnd מקור	red	12
0,1,2,3	אין אופrnd מקור	prn	13
אין אופrnd יעד	אין אופrnd מקור	rts	14
אין אופrnd יעד	אין אופrnd מקור	stop	15

### אלגוריתם שלדי של האסטמבלר

לחידוד ההבנה של תחליק העבודה של האסטמבלר, נציג להלן אלגוריתם שלדי למעבר הראשון ולמעבר השני. لتשומת לב: אין חובה להשתמש דווקא באלגוריתם זה.

אנו מחלקים את קוד המכונה לשני אזורים, אזור ההוראות (code) ואזור הנתונים (data). לכל אזור יש מונה משלה, ונסמנם IC (МОНІЯ ХОРОАТОВ - Instruction-Counter (Instruction-Counter - DC) (МОНІЯ НАТУНОВ - Data-Counter)). נבנה את קוד המכונה כך שייתאים לטעינה ל זיכרון **החל מכתובת 0**.

בכל מעבר מתחילה לקרוא את קובץ המקור מהתחלה.

#### מעבר ראשון

- .1. אתחל IC  $\leftarrow 0$ , DC  $\leftarrow$  DC.
- .2. קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עברו ל-17.
- .3. האם השדה הראשון בשורה הוא סמל? אם לא, עברו ל-5.
- .4. הדלק דגל "יש הגדרת סמל".
- .5. האם זהה הנחיה לאחסון נתונים, כולם, האם הנחיה data.או mat.? אם לא, עברו ל-8.
- .6. אם יש הגדרת סמל (תוויות), הכנס אותו לטבלת הסמלים עם המאפיין data. ערכו יהיה DC. (אם הסמל כבר נמצא בטבלה יש להודיע על שגיאה).
- .7. זהה את סוג הנתונים, קודד אותם בזיכרון, ועדכן את מונה הנתונים DC בהתאם לאורכם. חזרו ל-2.
- .8. האם זו הנחיה extern. או הנחיה entry.? אם לא, עברו ל-11.
- .9. אם זהה הנחיה entry. חזרו ל-2 (ההנחיה תטופל במעבר השני).
- .10. אם זו הנחיה extern., הכנס את הסמל המופיע כאופrnd של ההנחיה בתוך טבלת הסמלים עם הערך 0, ועם המאפיין external. חזרו ל-2.
- .11. זהה שורת הוראה. אם יש הגדרת סמל, הכנס אותו לטבלת הסמלים עם המאפיין code. ערכו של הסמל יהיה IC (אם הסמל כבר נמצא בטבלה יש להודיע על שגיאה).
- .12. חפש את שם הפעולה בטבלת שמות הפעולות, ואם לא נמצא, אז הודיע על שגיאה בשם ההוראה.

13. נתח את מבנה האופרנדים של ההוראה, וחשב מהו מספר המיללים הכלול בשתופותה ההוראה בקוד המכונה (נקרא למספר זה L).
14. בנה כתע את הקוד הבינארי של המילה הראשונה של ההוראה, ושל כל מילת-מידע נוספת שמוספת המוקודדת אופרנד במייעון מיידי.
15. שמור את הערכיים IC ו-L יחד עם נתוני קוד המכונה של ההוראה.
16. עדכן L  $\leftarrow IC + L$ , וחוור ל-2.
17. קובץ המקור נקרא בשלהותו. אם נמצא שגיאות במעבר הראשון, עזרו כאן.
18. שמור את הערכיים הסופיים של IC ושל DC (נקרא להם ICF ו-DCF). השתמש בהם לבניית קבצי הפלט, אחרי המעבר השני.
19. עדכן בטבלת הסמלים את ערכו של כל סמל המאופיין כ-data, ע"י הוספה הערך ICF (ראו הסבר לכך בהמשך).
20. התחל מעבר שני.

#### **מעבר שני**

1. קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עבור ל-7.
2. אם השדה הראשון בשורה הוא סמל (טווית), דרג עליו.
3. האם זהה הנחיתת.data או .mat או .string? אם כן, חזרו ל-1.
4. האם זהה הנחיתת.entry? אם לא, עבור ל-6.
5. הוסף בטבלת הסמלים את המאפיין entry למאפייני הסמל המופיע כאופרנד של ההנחיתת (אם הסמל לא נמצא בטבלת הסמלים, יש להודיע על שגיאיה). חזרו ל-1.
6. השלים את הקידוד הבינארי של מילוט-המידע של האופרנדים, בהתאם לשיטות המייעון שבשימוש. לכל אופרנד בקוד המקור המכיל סמל, מצא את ערכו של הסמל בטבלת הסמלים (אם הסמל לא נמצא בטבלה, יש להודיע על שגיאיה). אם הסמל מאופיין.external, הוסף את כוננות מילת-המידע הרלוונטי לרשימת מילוט-המידע שמתיחסות לסמל חיצוני. לפי ההוראות, לחישוב הקידוד והכתובות, אפשר להיעזר בערכיים IC ו-L של ההוראה, כפי שנשמרו במעבר הראשון. חזרו ל-1.
7. קובץ המקור נקרא בשלהותו. אם נמצא שגיאות במעבר השני, עזרו כאן.
8. בנה את קבצי הפלט (פרטים נוספים בהמשך).

נפעיל אלגוריתם זה על תוכנית הדוגמה שראינו לעיל (לאחר שלב פרישת המקראים), ונציג את הקוד הבינארי שמתתקבל במעבר ראשון ובמעבר שני. להלן שוב תכנית הדוגמה:

```

MAIN:    mov   M1[r2][r7],LENGTH
          add   r2,STR
LOOP:    jmp   END
          prn   #-5
          sub   r1, r4
          inc   K

          mov   M1[r3][r3],r3
          bne   LOOP
END:     stop
STR:    .string "abcdef"
LENGTH: .data  6,-9,15
K:      .data  22
M1:    .mat [2][2] 1,2,3,4

```

בוצע עתה מעבר ראשון על הקוד הנתון. בנה את טבלת הסמלים. כמו כן,בוצע במעבר זה גם את קידוד כל הנתונים, וקידוד המילה הראשונה של כל ההוראה. כמו כן, נקודד מילוט-המידע נוספת שמוספת של כל ההוראה, ככל שקידוד זה אינו תלוי בערך של סמל. את החלקים שעדיין לא מתורגמים במעבר זה, נשאיר כמוותיהם (מסומנים ב- ? בדוגמה להלן). נניח שהקוד יית�ע החל מהמען 100 (בבסיס 10).

<i>Label</i>	<i>Decimal Address</i>	<i>Base 4 Address</i>	<i>instruction</i>	<i>Operands</i>	<i>Binary machine code</i>
<i>MAIN:</i>	<b>0100</b>	1210	<b>mov</b>	M1[r2][r7],LENGTH כתובת של M1 קידוד אוגרי האינדקסים במטריצה	<b>0000-10-01-00</b> ? <b>0010-0111-00</b> ?
	<b>0101</b>	1211			?
	<b>0102</b>	1212		כתובת של LENGTH קידוד אוגרי האינדקסים במטריצה	<b>0010-0111-00</b>
	<b>0103</b>	1213		כתובת של LENGTH	?
	<b>0104</b>	1220	<b>add</b>	r2, STR קידוד מספר האוגר כתובת של STR	<b>0010-11-01-00</b> <b>0010-0000-00</b> ?
	<b>0105</b>	1221			
	<b>0106</b>	1222			
<i>LOOP:</i>	<b>0107</b>	1223	<b>jmp</b>	END כתובת של END	<b>1001-00-01-00</b> ?
	<b>0108</b>	1230			
	<b>0109</b>	1231	<b>prn</b>	#-5 המספר -5	<b>1100-00-00-00</b> <b>11111011-00</b>
	<b>0110</b>	1232			
	<b>0111</b>	1233	<b>sub</b>	r1,r4 קידוד מספרי האוגרים	<b>0011-11-11-00</b> <b>0001-0100-00</b>
	<b>0112</b>	1300			
	<b>0113</b>	1301	<b>inc</b>	K כתובת של K	<b>0111-00-01-00</b> ?
	<b>0114</b>	1302			
	<b>0115</b>	1303	<b>mov</b>	M1[r3][r3],r3 כתובת של M1 קידוד אוגרי האינדקסים במטריצה	<b>0000-10-11-00</b> ? <b>0011-0011-00</b> <b>0000-0011-00</b>
	<b>0116</b>	1310			
	<b>0117</b>	1311			
	<b>0118</b>	1312			
	<b>0119</b>	1313	<b>bne</b>	LOOP כתובת של LOOP	<b>1010-00-01-00</b> ?
	<b>0120</b>	1320			
<i>END:</i>	<b>0121</b>	1321	<b>stop</b>		<b>1111-00-00-00</b>
<i>STR:</i>	<b>0122</b>	1322	<b>.string</b>	“abcdef”	<b>0001100001</b>
	<b>0123</b>	1323			<b>0001100010</b>
	<b>0124</b>	1330			<b>0001100011</b>
	<b>0125</b>	1331			<b>0001100100</b>
	<b>0126</b>	1332			<b>0001100101</b>
	<b>0127</b>	1333			<b>0001100110</b>
	<b>0128</b>	2000			<b>0000000000</b>
<i>LENGTH:</i>	<b>0129</b>	2001	<b>.data</b>	6,-9,15	<b>0000000010</b> <b>1111110111</b> <b>0000001111</b>
	<b>0130</b>	2002			
	<b>0131</b>	2003			
<i>K:</i>	<b>0132</b>	2010	<b>.data</b>	22	<b>0000010110</b>
<i>M1</i>	<b>0133</b>	2011	<b>.mat</b>	[2][2] 1,2,3,4	<b>0000000001</b> <b>0000000010</b> <b>0000000011</b> <b>0000000010</b>
	<b>0134</b>	2012			
	<b>0135</b>	2013			
	<b>0136</b>	2020			

טבלת הסמלים אחרי המעבר ראשון היא :

סמל	ערך (בבסיס עשרוני)	איפיון הסמל
MAIN	100	code
LOOP	107	code
END	121	code
STR	122	data
LENGTH	129	data
K	132	data
M1	133	

נכע עתה את המעבר השני. נשלים באמצעות טבלת הסמלים את הקידוד החסר במיללים המסומנות "?".  
הקוד הבינארי בקורסו הסופית כאן זהה לקוד שהוצג בתחילת הנושא "**אסמבLER עם שני מעברים**".

הערה: כאמור, האסמבLER בונה קוד מכונה כך שתיארים לטעינה לזמן החל מכתובת 100 (עשרוני).  
אם הטעינה בפועל (לצורך הרצת התוכנית) תהיה לכתובת אחרת, יידרשו תיקונים בקוד הבינארי  
בשלב הטעינה, שיוכנסו בעורת מידע נוסף שהאסמבLER מכין בקבצי הפלט (ראו בהמשך).

<i>Label</i>	<i>Decimal Address</i>	<i>Base 4 Address</i>	<i>Command</i>	<i>Operands</i>	<i>Binary machine code</i>
<i>MAIN:</i>	<b>0100</b>	1210	<b>mov</b>	M1[r2][r7],LENGTH	<b>0000-10-01-00</b>
	<b>0101</b>	1211		M1	<b>10000101-10</b>
	<b>0102</b>	1212		קידוד אוגרי האינדקסים במטריצה	<b>0010-0111-00</b>
	<b>0103</b>	1213		כתובת של LENGTH	<b>10000001-10</b>
<i>LOOP:</i>	<b>0104</b>	1220	<b>add</b>	r2, STR	<b>0010-11-01-00</b>
	<b>0105</b>	1221		קידוד מספר האוגר	<b>0010-0000-00</b>
	<b>0106</b>	1222		כתובת של STR	<b>01111010-10</b>
<i>END:</i>	<b>0107</b>	1223	<b>jmp</b>	END	<b>1001-00-01-00</b>
	<b>0108</b>	1230		END	<b>01111001-10</b>
<i>STR:</i>	<b>0109</b>	1231	<b>prn</b>	#-5	<b>1100-00-00-00</b>
	<b>0110</b>	1232		-5	<b>11111011-00</b>
<i>K:</i>	<b>0111</b>	1233	<b>sub</b>	r1,r4	<b>0011-11-11-00</b>
	<b>0112</b>	1300		קידודי מספרי האוגרים	<b>0001-0100-00</b>
<i>MI:</i>	<b>0113</b>	1301	<b>inc</b>	K	<b>0111-00-01-00</b>
	<b>0114</b>	1302		כתובת של K	<b>10000100-10</b>
<i>LENGTH:</i>	<b>0115</b>	1303	<b>mov</b>	M1[r3][r3],r3	<b>0000-10-11-00</b>
	<b>0116</b>	1310		M1	<b>10000101-10</b>
	<b>0117</b>	1311		קידוד אוגרי האינדקסים במטריצה	<b>0011-0011-00</b>
	<b>0118</b>	1312		קידוד מספר האוגר של היעד	<b>0000-0011-00</b>
<i>DATA:</i>	<b>0119</b>	1313	<b>bne</b>	LOOP	<b>1010-00-01-00</b>
	<b>0120</b>	1320		LOOP	<b>01101011-10</b>
<i>END:</i>	<b>0121</b>	1321	<b>stop</b>		<b>1111-00-00-00</b>
<i>STR:</i>	<b>0122</b>	1322	<b>.string</b>	“abcdef”	<b>0001100001</b>
	<b>0123</b>	1323			<b>0001100010</b>
	<b>0124</b>	1330			<b>0001100011</b>
	<b>0125</b>	1331			<b>0001100100</b>
	<b>0126</b>	1332			<b>0001100101</b>
	<b>0127</b>	1333			<b>0001100110</b>
	<b>0128</b>	2000			<b>0000000000</b>
<i>LENGTH:</i>	<b>0129</b>	2001	<b>.data</b>	6,-9,15	<b>0000000010</b>
	<b>0130</b>	2002			<b>1111110111</b>
	<b>0131</b>	2003			<b>0000001111</b>
<i>K:</i>	<b>0132</b>	2010	<b>.data</b>	22	<b>0000010110</b>
<i>MI:</i>	<b>0133</b>	2011	<b>.mat</b>	[2][2] 1,2,3,4	<b>0000000001</b>
	<b>0134</b>	2012			<b>0000000010</b>
	<b>0135</b>	2013			<b>0000000011</b>
	<b>0136</b>	2020			<b>00000000100</b>

בסוף המעבר השני, אם לא נתגלו שגיאות, האסמבLER בונה את קבצי הפלט (ראו בהמשך), שמכילים את הקוד הבינארי ומידע נוסף עבור שלבי הקישור והטעינה.

כאמור, שלבי הקישור והטעינה אינם למימוש בפרויקט זה, ולא נדון בהם כאן.

## קבצי קלט ופלט של האסמבלי

ב�行ה של האסמבלי, יש להעביר אליו באמצעות ארגומנטים של שורת הפקודה (command line arguments) רשימה של שמות קבצי מקור (אחד או יותר). אלו הם קבצי טקסט, וביהם תוכניות בתחריר של שפת האסמבלי שהוגדרה בממ'ין זה.

האסמבלי פועל על כל קובץ מקור בנפרד, ויוצר עבורו קבצי פלט כלהלן:

- קובץ `.asm`, המכיל את קוד המקור לאחר שלב קדם האסמבלי (לאחר פרישת המאקרוואים).
- קובץ `.object`, המכיל את קוד המוכנה.
- קובץ `.externals`, ובו פרטיטים על כל המיקומות (הכתובות) בקוד המוכנה בהם יש מילת-מידע שמקודדת ערך של סמל שהוצע בחיצוני (סמל שהופיע כאופרנד של הנחיה `extern` ומאופיין בטבלת הסמלים `-c`). (external).
- קובץ `.entries`, ובו פרטיטים על כל סמל שנמצא כנקודות כניסה (סמל שהופיע כאופרנד של הנחיה `entry` ומאופיין בטבלת הסמלים `-c`). (entry).

אם אין בקובץ המקור אף הנחיה `extern`, האסמבלי לא יוצר את קובץ הפלט מסוג `externals`.  
אם אין בקובץ המקור אף הנחיה `entry`, האסמבלי לא יוצר את קובץ הפלט מסוג `entries`.

שמות קבצי המקור חייבים להיות עם הסיומת `".as"`. למשל, השמות `x.as`, `y.as`, ו-`hello.as` הם שמות חוקיים. העברת שמות הקבצים הללו כארוגומנטים לאסמבלי נעשית לא ציון הסיומת.

לדוגמה: נניח שתוכנית האסמבלי שלנו נקראת `assembler`, אז שורת הפקודה הבאה:  
`assembler x y hello`

תրץ את האסמבלי על הקבצים: `.x.as`, `y.as`, `hello.as`:

שמות קבצי הפלט מבוססים על שם קובץ הקלט, כפי שהופיע בשורת הפקודה, בתוספת סיומת מתאימה: הסיומת `".am"` עבור קובץ לאחר פרישת מאקרו, הסיומת `".ob"` עבור קובץ ה-`object`, הסיומת `".ext"` עבור קובץ ה-`entries`, והסיומת `".ext"` עבור קובץ ה-`externals`.

לדוגמה, בהפעלת האסמבלי באמצעות שורת הפקודה: `x assembler y ext`. וכן קבצי פלט `x.ob` ו-`x.am` וכל שיש הנחיות `.entry` או `.extern` בקובץ המקור.  
אם אין מאקרו בקובץ המקור, אז קובץ `"x.am"` יהיה זהה לקובץ `"x.as"`.

## אופן פעולות האסמבלי

נורחיב כאן על אופן פעולות האסמבלי, בנוסף לאלגוריתם השודי שניתן לעיל.

האסמבלי מחזק שני מערכים, שיקראו להן מערך ההוראות ומערך הנתונים. מערכים אלו נוטנים למשה תמונה של זיכרונות המוכנה (גודל כל כניסה במערך זהה לגודלה של מילת מכונה (בסיסיות). במערך ההוראות מכניס האסמבלי את הקידוד של הוראות המוכנה שנקראו במהלך המעבר על קובץ המקור. במערך הנתונים מכניס האסמבלי את קידוד הנתונים שנקראו מקובץ המקור (שורות מסוג `.data`).

לאסמבלי יש שני מונחים: מונה ההוראות (IC) ומונה הנתונים (DC). מונחים אלו מצבעים על המקום הבא הפניו במערכות לעיל, בהטאמה. כשהתחיל האסמבלי לעבור על קובץ מקור, שני מונחים אלו מקבלים ערך התחלתי.

בנוסף יש לאסמבלי טבלה, אשר בה נאספות כל התוויות בהן נתקל האסמבלי במהלך המעבר על הקובץ. לטבלה זו קוראים טבלת סמלים (symbol-table). לכל סמל (תוויות) נשמרים שמו, ערכו, ומאפיינים שונים שצינו קודם, כגון המיקום (code) או אופן העדכון (external) (למשל `data`).

האסטמבלר קורא את קובץ המקור שורה אחר שורה, מחליט מהו סוג השורה (הערה, הוראה, הנחיה, או שורה ריקה) ופועל בהתאם.

.1. שורה ריקה או שורת הערה : האסטמבלר מתעלם מהשורה וועבר לשורה הבאה.

.2. שורת הוראה :

האסטמבלר מוצא מהי הפעולה, ומהן שיטות המיעון של האופרנדים. (מספר האופרנדים אותם הוא מחפש נקבע בהתאם להוראה אותה הוא מצא).

אם האסטמבלר מוצא בשורת ההוראה גם הגדרה של תווית, אזי התווית מוכנסת אל טבלת הסמלים. ערך התווית הוא IC, והמאפיין הוא code.

האסטמבלר קובע לכל אופרנד את ערכו באופן הבא :

- אם זה רגיסטר – האופרנד הוא מספר הרגיסטר.
- אם זו תווית (מיון ישיר) – האופרנד הוא ערך התווית כפי שמופיע בטבלת הסמלים (ייתכן והסמל טרם נמצא בטבלת הסמלים).
- אם זה התו # ואחריו מספר – האופרנד הוא המספר עצמו.
- אם זו שיטת מיון אחרת – ערכו של האופרנד נקבע לפי המפרט של שיטת המיעון (ראו תיאור שיטות המיעון לעיל)

קביעת שיטות המיעון נעשית בהתאם לתחביר של האופרנד, כפי שהוסבר לעיל בהגדרת שיטות המיעון. למשל, מספר מצין מיידי, תווית מצינית מיון ישר וכד'.

לאחר שהאסטמבלר ניתח את השורה והתליט לגבי הפעולה, שיטת מיון אופרנד המקור (אם יש), ושיטות מיון אופרנד היעד (אם יש), הוא פועל באופן הבא :

אם זהה פעולה בעלת שני אופרנדים, אזי האסטמבלר מכניסים למערך ההוראות, במקומות אליו מציביע מונה ההוראות IC, את קוד המילה הראשונה של ההוראה (בשיטה הייצוג של הוראות המכונה כפי שתואר קודם לכן). מילה זו מכילה את קוד הפעולה, ואת שיטות המיעון. בנוסף "משרין" האסטמבלר מקום במערך עבור המילים הנוספות הנדרשות עבור הוראה זו, ומגדיל את מונה ההוראות בהתאם. אם אחד או שני האופרנדים הם בשיטת מיון רגיסטר או מיידי, האסטמבלר מקובד כעת את המילים הנוספות הרלוונטיות למערך ההוראות.

אם זהה פעולה בעלת אופרנד אחד בלבד, כלומר אין אופרנד מקור, אזי הקידוד הינו זהה לעיל, פרט לסיביות של שיטות המיעון של אופרנד המקור במילה הראשונה, אשר יכולו תמיד 0, מכיוון שאין רלוונטיות לפעולה.

אם זהה פעולה ללא אופרנדים אזי תקודד רק המילה הראשונה (והיחידה). הסיביות של שיטות המיעון של האופרנדים יכilio 0.

אם בשורת ההוראה קיימת תווית, אזי התווית מוכנסת אל טבלת הסמלים תחת השם המתאים. ערך התווית הוא ערך מונה ההוראות לפני קידוד ההוראה.

.3. שורת הנחיה :

כאשר האסטמבלר קורא בקובץ המקור שורת הנחיה, הוא פועל בהתאם לסוג הנחיה, באופן הבא :

I. 'data'.  
האסטמבלר קורא את רשימת המספרים, המופיעה לאחר 'data', מכניס כל מספר אל מערך הנתונים, ומגדם את מצביע הנתונים DC באחד עבור כל מספר שהוכנס.

אם בשורה 'data.', יש תווית או מוכנסת לטבלת הסמלים. היא מקבלת את הערך של מונה הנתונים DC שלפניהם הכנסת המספרים למערך הנתונים. כן מסומן שההגדרה ניתנה בחלק הנתונים.

II.'.mat' .  
האסטמבלר קורא את רשימת המספרים של איתחול המטריצה (אם יشنו), מכניס כל מספר שנקרה אל מערך הנתונים, ומقدم את מצביע הנתונים באחד עבור כל מספר שהוכנס. אם המטריצה אינה מאותחלת בערכים, יוקצו תאים במערך בהתאם לגודל המטריצה, והם יאותחלו ל-0.

טיפול בתווית המופיעה בשורה, זהה לטיפול הנעשה בהנחיה 'data.'.

III.'.string' .  
טיפול ב-'string' דומה ל-'data.', אלא שקודם ascii של התווים הם אלו המוכנסים אל מערך הנתונים (כלתו בכינסה נפרדת). לאחר מכן מוכנס הערך 0 (המצין סוף מחזרות) אל מערך הנתונים. מונה הנתונים מוקדם באורך המחרוזת + 1 (גם האפס בסוף המחרוזת תופס מקום).

טיפול בתווית המוגדרת בשורה זו זהה לטיפול הנעשה בהנחיה 'data.'

IV.'.entry' .  
זהי בקשה לאסטמבלר להכניס את התווית המופיעה כאופrnd של 'entry.' אל קובץ ה-entries. האסטמבלר רושם את הבקשה ובסיום העבודה, התווית הניל תירשם בקובץ ה-entries.

V.'.extern' .  
זהי הקריאה על סמל (תווית) המוגדר בקובץ אחר, ואשר קטע האסטמבלி בקובץ הנוכחי עושה בו שימוש. האסטמבלר מכניס את הסמל אל טבלת הסמלים. ערכו הוא 0 (הערך האמייתי לא ידוע, וייקבע רק בשלב הקישור), וטיפולו הוא external. לא ידוע באיזה קובץ נמצא הגדרת הסמל (וגם אין זה משנה עבור האסטמבלר).

יש לשים לב: בחוראה או בהנחיה אפשר להשתמש בשם של סמל אשר ההערה עליו ניתנת בהמשך הקובץ (אם באופן ישר על ידי הגדרת תווית, ואם באופן עקיף על ידי הנחיתת extern).

בסוף המעבר הראשון, האסטמבלר מעדכן בטבלת הסמלים כל סמל המופיע כ-'data', על ידי הוספת IC+100 (עשרוני) לערכו של הסמל. הסיבה לכך היא שבתמונה הכלולת של קוד המcona, הנתונים מופרדים מההראות, וכל הנתונים נדרשים להופיע אחרי כל ההראות. סמל מסווג data הוא למעשה תווית באזור הנתונים, והעדכו מוסיף לערך הסמל (כלומר כתובתו בזיכרון) את האורך הכלול של קידוד כל ההראות, בתוספת כתובת התחלת הטיעינה של הקוד, שהיא 100.

טבלת הסמלים מכילה כעת את כל הערכים הנחוצים להשלמת הקידוד (למעט ערכים של סמלים חיצוניים).

במעבר השני, האסטמבלר מבודד באמצעות טבלת הסמלים את כל המילים במערך ההראות שטרם קידדו במעבר הראשון. אלו הן מילים שצרכו להכיל כתובות של תוויות.

## פורמט קובץ ה-object

קובץ זה מכיל את תמונות הזיכרונו של קוד המוכונה, בשני חלקים: תמונות ההוראות ראשונה, ואחריה ובצמוד Tamonot hahtorot.

כזכור, האסטבלר מקובץ את ההוראות כך שתמונות ההוראות תתאים לטעינה החל מכתובת 100 (עשרוני) בזיכרונו. נשים לב שرك המעביר הראשון יודיעים מהו הגודל הכלול בתמונה ההוראות. מכיוון שתמונות הנתונים נמצאת אחרי Tamonot hahtorot, גודל Tamonot hahtorot משפיע על הכתובות בתמונות הנתונים. זו הסיבה שבגללה היה צריך לעדכן בטבלת הסמלים, בסוף המעביר הראשון, את ערכי הסמלים המאופיינים-data (כזכור, בצעד 19 הוספנו לכל סמל כזה את הערך ICF). במעבר השני, בהשלמת הקידוד של מילוט-המידע, משתמשים בערכים המעודכנים של הסמלים, המותאמים למבנה המלא והוסף של Tamonot hahtorot.

כעת האסטבלר יוכל לכתוב את Tamonot hahtorot בשלהותה לתוך קובץ פלט (קובץ ה-object).

עקרונית, קובץ object מכיל את Tamonot hahtorot שתווארה כאנו. קובץ object מורכב משורות של טקסט. השורה הראשונה מכילה שני מספרים: אורך כולל של קלטן ההוראות (בAMILOT ZYCRON) ואורך כולל של קלטן הנתונים (AMILOT ZYCRON). השורות הבאות מתארות את תוכן הזיכרונו. בכל שורה שני מספרים: כתובות של מילה בזיכרונו, ותוכן המילה. כל המספרים בקובץ object הם בסיס 4 "יחודי" שהוגדר לעיל.

במהשך מופיע קובץ object לדוגמא בשם ps.object המתאים לקובץ המקור as ps.as עברו כל תא זיכרון המכיל הוראה (לא נתוניים) מופיע בקובץ object מידע עבור תכנית הקישור. מידע זה ניתן ע"י 2 הסיביות הימניות של הקידוד (שדה ה-E,R,A).

האות 'A' (קייזר של absolute) מצינית שתוקן התא איינו תלוי במקומות בזיכרונו בו ייעטן בפועל קוד המוכונה של התכנית בזמן ביצועה למשל מילה המכילה אופrnd מדידי).

האות 'R' (קייזר של relocatable) מצינית שתוקן התא כן תלוי במקומות בזיכרונו שבו ייעטן בפועל קוד המוכונה של התכנית בזמן ביצועה. לכן יש לעדכן את תוכן התא, בשלב הטיענה, על ידי הוספה היסט (Offset) מתאים (ההיסטוריה זה המعن בו תפטע המילה הראשונה של התכנית). במקרה כזה 2 הסיביות הימניות יכולו את הערך 10

האות 'E' (קייזר של external) מצינית שתוקן התא תלוי בערכו של סמל חיצוני (external), וכי רק בזמן הקישור ניתן יהיה להכנס לטא את הערך המתאים. במקרה כזה 2 הסיביות הימניות יכולו את הערך 01.

## פורמט קובץ ה-entries

קובץ entries בנייתו משורות טקסט. כל שורה מכילה שם של סמל שהוגדר כ-entry ואת ערכו, כפי שנמצא בטבלת הסמלים. הערכים כוללים בסיס 4 הייחודי M1: .mat [2][2] 1,2,3,4 (ראו לדוגמה את הקובץ ent.ps.as המתאים לקובץ המktor שלו).

## פורמט קובץ ה-externals

קובץ externals בנייתו אף הוא משורות טקסט. כל שורה מכילה שם של סמל שהוגדר כ-external, וכותבת בקוד המוכונה בה יש קידוד של אופrnd המתיחס לSymbol זה. מבון שייתכן ויש מספר כתובות בקוד המוכונה בהם מתיחסים לאותו סמל חיצוני. לכל התייחסות כזו תהיה שורה נפרדת בקובץ externals. הכתובות מיוצגות בסיס 4 הייחודי.

لتשומת לב: ניתן שיש מספר כתובות בקוד המוכונה בהן מילוט-המידע מתייחסות לאותו סמל חיצוני. ככל כתובות כזו תהיה שורה נפרדת בקובץ externals.

נדגים את קבצי הפלט שמייצר האסמבילר עבור קובץ מקור בשם ps.as.  
התוכנית לאחר שלב פרישת המאקרו תיראה כך:

```
; file ps.as

.entry LOOP
.entry LENGTH
.extern L3
.extern W
MAIN:    mov   M1[r2][r7],W
          add   r2,STR
LOOP:     jmp   W
          prn   #-5
          sub   r1, r4
          inc   K

          mov   M1[r3][r3],r3
          bne   L3
END:      stop
STR:      .string "abcdef"
LENGTH:   .data  6,-9,15
K:        .data  22
M1:      .mat [2][2] 1,2,3,4
```

להלן טבלת הקידוד הבינארי המלא שמתקבל מקובץ המקור, כפי שנבנה מעבר הראשוני והשני.

Label	Decimal Address	Base 4 Address	Instruction	Operands	Binary machine code
MAIN:	0100	1210	<b>mov</b>	M1[r2][r7],W	<b>0000-10-01-00</b>
	0101	1211		כתובת של M1	<b>10000101-10</b>
	0102	1212		קידוד אוגרי האינדקסים במטריצה	<b>0010-0111-00</b>
	0103	1213		כתובת של W (סמל חיצוני)	<b>00000000-01</b>
	0104	1220	<b>add</b>	r2, STR	<b>0010-11-01-00</b>
	0105	1221		קידוד מספר האוגר	<b>0010-0000-00</b>
	0106	1222		כתובת של STR	<b>01111010-10</b>
LOOP:	0107	1223	<b>jmp</b>	W	<b>1001-00-01-00</b>
	0108	1230		כתובת של W	<b>00000000-01</b>
	0109	1231	<b>prn</b>	#-5	<b>1100-00-00-00</b>
	0110	1232		-5 המספר	<b>11111011-00</b>
	0111	1233	<b>sub</b>	r1,r4	<b>0011-11-11-00</b>
	0112	1300		קידודי מספרי האוגרים	<b>0001-0100-00</b>
	0113	1301	<b>inc</b>	K	<b>0111-00-01-00</b>
	0114	1302		כתובת של K	<b>10000100-10</b>
	0115	1303	<b>mov</b>	M1[r3][r3],r3	<b>0000-10-11-00</b>
	0116	1310		כתובת של M1	<b>10000101-10</b>
	0117	1311		קידוד אוגרי האינדקסים במטריצה	<b>0011-0011-00</b>
	0118	1312		קידוד מספר האוגר של היעד	<b>0000-0011-00</b>
	0119	1313	<b>bne</b>	L3	<b>1010-00-01-00</b>
	0120	1320		כתובת של L3 (סמל חיצוני)	<b>00000000-10</b>
END:	0121	1321	<b>stop</b>		<b>1111-00-00-00</b>

<i>STR:</i>	<b>0122</b>	1322	<b>.string</b>	“abcdef”	<b>0001100001</b>
	<b>0123</b>	1323			<b>0001100010</b>
	<b>0124</b>	1330			<b>0001100011</b>
	<b>0125</b>	1331			<b>0001100100</b>
	<b>0126</b>	1332			<b>0001100101</b>
	<b>0127</b>	1333			<b>0001100110</b>
	<b>0128</b>	2000			<b>0000000000</b>
<i>LENGTH:</i>	<b>0129</b>	2001	<b>.data</b>	6,-9,15	<b>0000000110</b>
	<b>0130</b>	2002			<b>1111110111</b>
	<b>0131</b>	2003			<b>0000001111</b>
<i>K:</i>	<b>0132</b>	2010	<b>.data</b>	22	<b>0000010110</b>
<i>M</i>	<b>0133</b>	2011	<b>.mat</b>	[2][2] 1,2,3,4	<b>0000000001</b>
	<b>0134</b>	2012			<b>0000000010</b>
	<b>0135</b>	2013			<b>0000000011</b>
	<b>0136</b>	2020			<b>0000000100</b>

### הקובץ ob.ps

**כל תוכן הקובץ מיוצג במספריים בסיס 4 הייחודי.**  
**הערה : שורת הכוורת אינה חלק מהקובץ, ונועדה להבירה בלבד.**

Base 4 address    Base 4 code

bbc	dd
bcba	<b>aacba</b>
bcbb	<b>cabbc</b>
bcbc	<b>acbda</b>
bcbd	<b>aaaab</b>
bcca	<b>acdba</b>
bccb	<b>acaaa</b>
bccc	<b>bdecc</b>
bcdd	<b>cbaba</b>
bcda	<b>aaaab</b>
bcd <sub>b</sub>	<b>daaaa</b>
bedc	<b>ddcda</b>
bcdd	<b>addda</b>
bd <sub>a</sub> a	<b>abbaa</b>
bdab	<b>bdaba</b>
bdac	<b>cabac</b>
bdad	<b>aacda</b>
bdba	<b>cabbc</b>
bdbb	<b>adada</b>
bd <sub>b</sub> c	<b>aaada</b>
bd <sub>b</sub> d	<b>ccaba</b>
bdca	<b>aaaab</b>
bdcb	<b>ddaaa</b>
bdcc	<b>abcab</b>
bdcd	<b>abcac</b>
bdda	<b>abcd</b>
bddb	<b>abeba</b>
bddc	<b>abeb<sub>b</sub></b>
bddd	<b>abebc</b>
caaa	<b>aaaaa</b>
caab	<b>aaabc</b>
caac	<b>dddbd</b>
caad	<b>aaadd</b>
caba	<b>aabbc</b>
cabb	<b>aaaab</b>
cabc	<b>aaaac</b>
cabd	<b>aaaad</b>
caca	<b>aaaba</b>

### קובץ ent.ps

LOOP                bcc<sub>d</sub>  
 LENGTH              caab

### קובץ ext.ps

W                bcbd  
 W                bcda  
 L3                bdca

**לשומת לב :** אם בקובץ המקור אין הנחיות `extern`. אז לא ייווצר קובץ `ext`. בדומה, אם אין בקובץ המקור הנחיות `entry`., לא ייווצר קובץ `ent`. **אין לייצור קובץ ext או ent שנשאר ריק.** הערה : אין חשיבות לסדר השורות בקבצים מסווג `ext`. או `ent`. כל שורה עומדת בפני עצמה.

## סיכום והנחיות כלליות

- אורך התוכנית, הניתנת כקלט לאסטבלר אינו ידוע מראש, ולכן אורך התוכנית המתורגמת אינו אמור להיות צפוי מראש. אולם כדי להקל בימוש האסטבלר, ניתן להניח גודל מקסימלי. לפיקד יש אפשרות להשתמש במערכיים לאחסון תमונת קוד המcona **בלבד**. כל מבנה נתונים אחר (למשל טבלת הסמלים וטבלת המאקרו), יש למשם באופן יעיל וחסכוני (למשל באמצעות רישימה מקושרת והקצת זיכרון דינמי).
  - השימוש של קבצי הפלט צריכים להיות תואמים לשם קובץ הקלט, למעט הסיומות. למשל, אם קובץ הקלט הוא `prog.as` אז קבצי הפלט שייצרו הם : `prog.ob`, `prog.ext`, `prog.ent`.
  - מתכונות הפעלת האסטבלר צריכה להיות כפי הנדרש בממ"ן, ללא שינוים כלשהם. ככלומר, משחק המשמש יהיה אך ורק באמצעות שורת הפקודה. בפרט, שמות קבצי המקוור יועברו לתכנית האסטבלר כארגומנטים בשורת הפקודה. אין להוסיף תפריטי קלט אינטראקטיביים, חלונות ורפיים למיניהם, ועוד'.
  - יש להקפיד לחלק את השימוש האסטבלר למספר מודולים (קבצים בשפת C) לפי MISIMOT. אין לרכו MISIMOT מסווגים שונים במודול יחיד. מומלץ לחלק למודולים כגון: מעבר ראשון, מעבר שני, פונקציות עזר (למשל, תרגום לבסיס, ניתוח תחבירי של שורה), טבלת הסמלים, מפת הזיכרון, טבלאות קבועות (קודם הפעלה, שיטות המיעון החוקיות לכל פעולה, ועוד').
  - יש להקפיד ולתעד את השימוש באופן מלא וברור, באמצעות הערות מפורחות בקוד.
  - יש לאפשר תווים לבנים עודפים בקובץ הקלט בשפת אסטבלר. למשל, אם בשורת הוראה יש שני אופרנדים המופרדים בפסיק, אין לפניו ואחריו הפסיק מותר שהיהו רוחניים וטאבבים בכל מקרה. בדומה, גם לפניו ואחריו שם הפעולה. מוגדרות גם שורות ריקות. האסטבלר יתעלם מהווים לבנים מיותרים (כלומר ידלג עליהם).
  - הקלט (קוד האסטבלרי) עלול להכיל שגיאות תחביריות. על האסטבלר לגלות ולדוח על כל השורות השגויות בקלט. **אין לעצור את הטיפול בקובץ קלט לאחר גילוי השגיאה הראשונה.** יש להזפיס למשך הודעות מפורחות ככל הניתן, כדי שאפשר יהיה להבין מה והיכן כל שגיאה. כמובן שגם קובץ קלט מכיל שגיאות, אין טעם להפסיק עבورو את קבצי הפלט (`ob`, `ext`, `ent`).
- תשנו ונסלם פרק ההסבירים והגדרת הפרויקט.**
- בשאלות ניתן לננות לקבוצת הדיוון באתר הקורס, ועל המנחים בשעות הקבלה.**
- lezicircums, מומלץ לכל אלה שטרם בדקו את התכנים באתר הקורס לעשות זאת. נשאלות באתר הרבה שאלות בנושא חומר הלימוד והממי"נים, והתשובה יכולות להועיל לכם.
- לשומת לבכם :
- על המטלה להיות מקורית למורי: אין להיעזר בספריות חיצונית מלבד הספריות הסטנדרטיות, וכמובן לא בקוד שמצאתם בראש או קיבתם בכלל זהך. אין לשתף בראש קוד אלא סיסמה. אלו הן **עבירות ממשעת**.
  - לא תיתנו דחיה בהגשת הממי"ן, פרט למקרים חריגים **במיוחד**, כגון **אישפו ממושך**. במקרה אלו יש לבקש ולקבל אישור מראש **מנחתה הקבוצה**.

בזה צלח !