

# מטלת מנהה (ממ"ז) 14

הקורס : 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטרלה : פרויקט גמר

משקל המטרלה : 31 נקודות (חוובה)

מספר השאלות : 1

מועד אחרון להגשה : 12.8.2019

סמסטר : 2019/ב'

## קיימות שתי חלופות להגשת מטלות:

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס

- שליחת מטלות באמצעות דואר אלקטרוני - באישור המנהה בלבד

### הסבר מפורט ב"נווה הגשת מטלות מנהה"

אחת 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 ל כתוב תוכנת אסמבילר, עבור שפת אסמביל שתוכדר בהמשך. הפרויקט יכתב בשפת C

עליכם להגיש :

1. קבצי המקור של התוכנית שתכתבם (קבצים בעלי סיווגת c. או h.).

2.קובץ הרצה (מקומפל ומקשור) עבור מערכת אוביונטו.

3. קובץ makefile. יש להשתמש בкомפיילר gcc עם הדגמים : -Wall -ansi -pedantic . יש לנפות את כל ההודעות שימושה הקומפיילר, כך שהתוכנית תתקמפל ללא כל העורות או אזהרות.

4. דוגמאות הרצה (קלט ופלט) :

**א.** קובצי קלט בשפת אסמביל, וקובצי הפלט שנוצרו מהפעלת האסמביל על קבצי קלט אלה. יש להציג שימוש במגוון הפעולות וטיפוסי הנתונים של שפת האסמביל.

**ב.** קובצי קלט בשפת אסמביל המדגים מגוון רחב של סוגים שגיאות אסמביל (ולבן לא נוצרים קבצי פלט), ותדריסי המשך המראים את ה הודעות השגיאה שמוציאה האסמביל.

בשל גודל הפרויקט, עליכם לחלק את התוכנית למספר קבצי מקור, לפי מישימות. יש להזכיר שקוד המקור של התוכנית יעמוד בקריטריונים של בהירות, קריאות וכתייה נאה ו邏輯ית.

נזכיר מספר היבטים חשובים של כתיבת קוד טוב :

1. הפיטה של בני הנטונים : רצוי (כל האפשר) להפריד בין הגישה לבני הנטונים לבני המימוש של בני הנטונים. כך, למשל, בעת כתיבת פונקציות לטיפול בטבלה, אין זה מעניינים של משתמשים בפונקציות אלה, האם הטבלה ממושחת באמצעות מערך או באמצעות רשימה מקוורת.

2. קריאות הקוד : יש להשתמש במסמות משמעותיים למשתנים ופונקציות. כמו כן, רצוי להגדיר קבועים רלוונטיים תוך שימוש בהנחת #define, ולהימנע מ"מספרים קשים", שימושות נחרה לכם בלבד. יש לערך את הקוד באופן מסודר: הזחות עיקריות, שורות ריקות להפרדה בין קטעי קוד, וכו'.

3. תיעוד : יש להכנס בקבצי המקור תיעוד תמציתי וברור, שיסביר את תפקידה של כל פונקציה (באמצעות הערות כותרת לכל פונקציה). כמו כן יש להסביר את תפקידם של משתנים חשובים. כמו כן, יש להכנס הערות ברמת פירוט גבוהה בכל הקוד.

הערה: תוכנית "עובדת", דהיינו תוכנית שמבצעת את הדרוש ממנה, אינה לכשעצמה ערובה לציון גבוה. כדי לקבל ציון גבוה, על התוכנית לעמוד בקריטריונים של כתיבה ותיעוד ברמה טובה, כמפורט לעיל, אשר משקלם המשותף מגיעה עד לכ- 40% משקל הפרויקט.

יותר להשתמש בפרויקט בכל מגוון הספריות הסטנדרטיות של שפת C, אבל אין להשתמש בספריות חיצונית אחרות.

מומלץ לעבוד בזוגות. אין לעבוד בצוותים גדולים יותר. **פרויקט שיוגש על ידי שלשה או יותר, לא יבדק ולא יקבל ציון**. חוברים שהגיעו יחד את הפרויקט, יהיו **שייכים** לאותה **קבוצת הנחיה**. הציון יהיה זהה לשני הסטודנטים.

מומלץ לקרוא את הגדרת הפרויקט פעמיים ראשונה ברכז, לקבלת תמונה 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 לרוץ באותו מחשב עצמו. כיצד "מכיר" המחשב כל כך הרבה שפות? התשובה פשוטה: המחשב מכיר למעשה שפה אחת בלבד: הוראות ונתונים הכתובים בקוד ביארי. קוד זה מאוחסן בגוש זיכרון, ונראה כמו רצף של ספרות בינאריות. יחידת העיבוד המרכזית - היע"מ (CPU) - יודעת לפרק את הרץ הזה לקטעים קטנים בעלי משמעות: הוראות, מענים ונתונים.

למעשה, זיכרון המחשב יכול הוא אוסף של סיביות, שנוהגים לראותו כמרקיבות ליחידות בעלות אורך קבוע (בתים, מילימטרים). לא ניתן להבחין, בין שאינה מיומנת, בהבדל פיסי כלשהו בין אותו חלק בזכרון שבו נמצאת תוכנית לבין שאר הזיכרון.

יחידת העיבוד המרכזית (היע"מ) יכולה לבצע מגוון פעולות פשוטות, הנקראות **הוראות מכונה**, ולשם כך היא משתמשת באוגרים (registers) הקיימים בתוך היע"מ, ובזכרון המחשב. **דוגמאות**: העברת מספר מתא בזכרון לאוגר ביע"מ או בחזרה, הוספה 1 למספר הנמצא באוגר, בדיקה האם המאוחסן באוגר שווה לאפס, חיבור וחיסור בין שני אוגרים, וכו'. הוראות המכונה ושילובים שלן הן המרכיבות תוכנית כפי שהיא טעונה לזרן בזמן ריצתה. כל תוכנית מקור (התוכנית כפי שנכתבה בידי המתכנן), תתרגם בסופו של דבר באמצעות תוכנה מיוחדת לזרה סופית זו.

היע"מ יודע לבצע קוד שנמצא בפורמט של **שפת מכונה**. זהו רצף של ביטים, המהווים קידוד ביןארי של סדרת הוראות המכונה המרכיבות את התוכנית. קוד זה אינו קריא למשתמש, ולכן לא נוח לקודד (או לקרוא) תכניות ישירות בשפת מכונה. **שפת אסמלבי** (assembly language) היא שפת תוכנות מאפשרת לייצג את הוראות המכונה בזרה סימבולית קלה ונוחה יותר לשימוש. כמו כן יש צורך לתרגם את הייצוג הסימבולי לקוד בשפת מכונה, כדי שהתוכנית תוכל לרוץ במחשב. תרגום זה נעשה באמצעות כלי שנקרא **אסמלבלר** (assembler).

כידוע, לכל שפת תוכנות עילית יש מהדר (compiler), או מפרש (interpreter), המתרגם תוכניות מקור לשפת מכונה. האסמלבלר משמש בתפקיד דומה עבור שפת אסמלבי.

כל מודל של יע"מ (כלומר לכל אירוגון של מחשב) יש שפת מכונה יעודית משלו, ובהתאם גם שפת אסמלבי יעודית משלו. לפיכך, גם האסמלבלר (כלי התרגומים) הוא יעודית ושונה לכל יע"מ.

תפקידו של האסמלבלר הוא לבנות קוד המכיל קוד מכונה, מקובץ נתון של תוכנית הכתובה בשפת אסמלבי. זהו השלב הראשון במסלול אותו עברת התוכנית, עד לקבלת קוד המוכן לריצה על חומרת המחשב. השלבים הבאים הם קישור (linkage) וטעינה (loading), אך בהם לא נעסק במעטן זה.

המשימה בפרויקט זה היא לכתוב אסמלבלר (כלומר תוכנית המתרגם לשפת מכונה), עבר שפת אסמלבי שנגידר כאן במיוחד לצורך הפרויקט.

**لتשומת לב**: בהסברים הכלליים על אופן עבודה תוכנת האסמלבלר, 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 האסמלבלר. אין לטעות: עליהם לכתוב את תוכנית האסמלבלר בלבד. אין לכתוב את תוכניות הקישור והטעינה!!!

## המחשב הדמיוני ושפת האסטבלי

נגידר עתה את שפת האסטבלי ואת מודל המחשב הדמיוני, עבור פרויקט זה.  
הערה : תאור מודל המחשב להלן הוא חלקו בלבד, ככל שנחוץ לביצוע המשימות בפרויקט.

"חומרה":

המחשב בפרויקט מורכב ממעבד (יע"מ), אוגרים (רגיסטרים), זיכרון RAM. חלק מהזיכרון משמש כמחסנית (stack).

למעבד 8 אוגרים כליליים, בשמות : r0, r1, r2, r3, r4, r5, r6, r7. גודלו של כל אוגר הוא 14 סיביות. הסיבית ה-<sup>ci</sup> פחותה משמעותית מצוין כסיבית מס' 0, והסיבית המשמעותית ביותר במס' 13. שמות האוגרים נכתבים תמיד עם אות 'z' קטנה.

כמו כן יש במעבד אוגר 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 4096 תאים, בכתביות 4095-0 (בסיס עשרוני), וכל תא הוא בגודל של 14 סיביות. לתא בזכרון נקרא גם בשם "מילה". הסיביות בכל מילה ממושפרות כמו באוגר.

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

מבנה הוראות מכונה:

כל הוראה מכונה מקודדת במספר מילوت זיכרון רצופות, החל ממילה אחת ועד למקסימום חמיש מילים, הכל בהתאם לשיטות המיון בהן נעשה שימוש (ראו בהמשך).

בקובץ הפלט המכיל את קוד המכונה שבונה האסטבלי, כל מילה תקודד בסיס 4 ("מיוחד" המוגדר כדלקמן (ראו הסברים ודוגמאות בהמשך) :

בסיס 4 רגיל	0	1	2	3
בסיס 4 מיוחד	*	#	%	!

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

A,R,E	מיוען אופrnd יעד	מיוען אופrnd מקור	opcode	לא בשימוש	13 12 11 10
-------	------------------	-------------------	--------	-----------	-------------

**סיביות 9-6**: במילה הראשונה של ההוראה סיביות אלה מהוות את **קוד-הפעולה** (opcode). כל opcode מיוצג בשפת אסטבלי באופן סימבולי על ידי **שם-פעולה**.

בשפה שלנו יש 16 קודים פעולה והם :

שם הפעולה	קוד הפעולה (בסיס עשרוני)
mov	0
cmp	1
add	2
sub	3
not	4
clr	5
lea	6
inc	7
dec	8
jmp	9
bne	10
red	11
prn	12
jsr	13
rts	14
stop	15

שמות הפעולות נכתבים תמיד באותיות קטנות. פרוטו המשמעות של הפעולות יבוא בהמשך.

**סיביות 2-3:** מקודדות את מספרה של שיטת המיעון של אופרנד היעד (destination operand) . אם אין בהוראה אופרנד מקור, ערךן של סיביות אלה הוא 0.

**סיביות 5-4:** מקודדות את מספרה של שיטת המיעון של אופרנד המקור (source operand) . אם אין בהוראה אופרנד יעד, ערךן של סיביות אלה הוא 0.

**סיביות 13-10:** אין בשימוש וערךן הוא 0.

**סיביות 1-0 (השדה 'E,R,A'):** במילה הראשונה של הוראה סיביות אלה תמיד מאופסות (00).

לתשומת לב : **השדה 'E,R,A'** מותוסף לכל אחת מהמילים בקידוד ההוראה (ראו פירוט שיטות המיעון להלן).

#### שיטות מיעון:

בשפת האסמבלי שלנו קיימות ארבע שיטות מיעון, המסומנות במספרים 0,1,2,3. השימוש בשיטות מיעון מצרך קידוד של מילות-מידע נוספות בקוד המכונה של כל הוראת מכונה. כל אופרנד של ההוראה תופס מילה אחת או שתי מילים נוספות, תלוי בשיטת המיעון של האופרנד.

כאשר בהוראה יש שני אופרנדים, קודם יופיעו מילות-המידע הנוספות של האופרנד הראשון (אופרנד המקור), ולאחר מכן מילות-המידע הנוספות של האופרנד השני (אופרנד היעד). קיימים גם מקרה מיוחד בו קידוד שני האופרנדים נעשה באמצעות מילת-מידע אחת משותפת לשני האופרנדים.

כאמור, בכל מילת-מידע נוספת של ההוראה, סיביות 1-0 הן השדה מצין מהו סוג הקידוד של המילה : קידוד מוחלט (Absolute) , חיצוני (External) או ניתן להזזה (Relocatable). הערך 00 ממשמעות שקידוד המילה הוא מוחלט (ואינו מצרך שינוי בשלבי הקישור והטיענה). הערך 01 ממשמעות שהקידוד הוא של כתובת חיצונית (ומצריך שינוי בשלבי הקישור והטיענה). הערך 10 ממשמעות שהקידוד הוא של כתובת פנימית ניתנת להזזה (ומצריך שינוי בקישור ובטיענה).

אפשרו יותר מפורט של תפקיד השדה 'E,R,A' בקוד המכונה מופיע בהמשך.

להלן תיאור שיטות המיעון:

ערך	שיטת מיון	תוכן המילה נוספת	אופן הכתיבה	דוגמה
0	מיון מיידי	מילת-מיון נוספת של ההוראה מכילה את האופrnd עצמו, שהוא מספר שלם בשיטת המילים ל-2, המיצג ברוחב של 12 סיביות, אליהן מתווספות זוג סיביות של השדה A,R,E (הערך של שדה זה הוא תמיד 00 עבור מיון מיידי).	האופrnd מתחילה בתו # ולאחריו ובצמוד אליו מופיע מספר שלם בסיס עשרוני. יש גם אפשרות שבמקרים מסווגר יופיע שם של מקאו שהוגדר בתכנית (ראו פרטים בהמשך).	mov #-1,r2  בדוגמה זו האופrnd הראשון של הפקודה נתון בשיטת מיון מיידי. ההוראה כתובת את הערך -1 אל אוגר r2  דוגמה נוספת: נתונה הגדרת המאקרו:.define size = 8  zioni ההוראה: mov #size, r1  האופrnd הראשון הוא 8 מיידי, כאשר המספר 8 מיוצג באמצעות שם המאקרו size. ההוראה כתובת את הערך 8 אל אוגר r1
1	מיון ישיר	מילת-מיון נוספת של מילה בזיכרון. מילה זו בזיכרון היא האופrnd. הכתובת מיוצגת כמספר ללא סימן ברוחב של 12 סיביות, אליהן מתווספות זוג סיביות של השדה A,R,E (הערך של שדה זה הוא או 01 או 10, תלוי בסוג הכתובת - חיונית או פנימית).	האופrnd הוא <u>תוויות</u> שכבר הוצאה או שתוצאה בהמשך הקובץ. הכתובת נעשית על ידי כתיבת <u>תוויות</u> בתחילת הנקיה '.string' או '.data' או '.extern'. או בתחילת הוראה של התוכנית, או באמצעות אופrnd של הנקיה 'ז'ו'.	נתונה ההגדרה: x: .data 23  zioni ההוראה: dec x  מקטינה ב-1 את תוכן x המילה שבכנתובת x בזיכרון ("משתנה" x).
2	מיון אינדקס קבוע	שיטת מיון זו משמשת לגישה לאייר במערך לפי אינדקס. המערך נמצא בזיכרון. כל אייר במערך הוא בגודל מילה.  בשיטת מיון זו קיימות בקידוד ההוראה שתי מילוט-מיון נוספות. המילה הנוספת הראשונה מכילה את כתובות התחלה המערך. המילה הנוספת השנייה מכילה את הנקודות המערך. האינדקס של האיר במערך אליו יש לגשת.  הערכים בשתי מילוט-מיון הנוספות מיוצגים ברוחב 12 סיביות, אליהן מתווספות זוג סיביות של השדה A,R,E (הערך של שדה זה במילוט-מיון הרשונה הוא כמו במילוט-מיון ישר, ובמילת-מיון השני כמו במילוט-מיון מיידי).	האופrnd מרכיב מtwoites המציג את כתובות התחלה המערך, ולאחריה בטוראים מרובעים האינדקס במערך אליו רוצים לפנות.  האינדקס יכול להינתן ע"י קבוע מספרי, או ע"י שם של מקאו שהוגדר בעוזרת define. (יוסבר בהמשך). האינדקסים במערך מתחילה מ-0.	נתונה ההגדירה: x: .data 23,25,19,30  zioni ההוראה: mov x[2],r2  תעתיק את המספר 19 הנמצא באינדקס 2 במערך x אל אוגר r2.  דוגמה נוספת: נתונה הגדרת המערך x עליל, וכן הגדרת המאקרו: .define k=1  zioni ההוראה: mov r2,x[k]  תעתיק את תוכן האוגר r2 אל המילה באינדקס 1 במערך x (נדרס התוכן הקודם (25).

ערך	שיטת מייען	תוכן המילה נוספת	אופן הכתיבה	דוגמה
3	מייען אוגר ישיר	<p>האופrnd הוא שם של אוגר.</p> <p>בדוגמה זו, הוראה מעתקה את תוכן אוגר r1 אל אוגר r2.</p> <p>בדוגמה זו, שני האופrndים הם בשיטת מייען אוגר ישיר, ולכן יחולקו מילת-מידע נספת אחת משותפת.</p>	<p>האופrnd הוא שם של אוגר.</p> <p>בנסיבות 2-4 את מספרו של האוגר. ואילו אם האוגר משתמש כאופrnd מקור, מספר האוגר יקודד בסביבות 7-5 של מילת-המידע.</p> <p>אם בפקודה יש שני אופrndים ושניהם בשיטת מייען אוגר ישיר, הם יחולקו מילת-מידע אחד משותפת, כאשר הסביבות 2-4 הן עבר אוגר היעד והסביבות 5-7 הן עבר אוגר המקור.</p> <p>לAMILT-MIDU מ��וקיפות זוג סיביות של השדה A,R,E (הערך של שדה זה הוא תמיד 00 עבר מייען אוגר ישיר).</p> <p>סיביות אחרות במילת-המידע שאינו בשימוש יובילו 0.</p>	<p>mov r1,r2</p> <p>בדוגמה זו, הוראה מעתקה את תוכן אוגר r1 אל אוגר r2.</p>

הערה : מותר להשתמש בתווית עוד לפני שמנדרים אותה, בתנאי שהתוויות אכן מוגדרות במקום כלשהו בהמשך הקובץ.

#### מפורט הוראות המכונה :

הוראות המכונה מתחולקות לשלווש קבוצות, לפי מספר האופrndים הדורש להן.

**קבוצת הוראות הראשונה :**  
אלן הוראות הדורשות שני אופrndים.

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

הוראה	הסבר הפעולה	דוגמה	הסבר הדוגמה
mov	מבצעת העתקה של האופrnd הראשון, אופrnd המקור (source) אל האופrnd השני, אופrnd היעד (destination) בהתאם לשיטת המייען.	mov A, r1	העתק את תוכן המשתנה A (הmillionה שבכבותה בזיכרונו) אל אוגר r1.
cmp	מבצעת "השוואה" בין שני האופrndים שלה. אופן ההשוואה: תוכן אופrnd היעד (השני) מופחת מתוכן אופrnd המקור (הראשון), ללא שבירת תוכנת החישור. פעולה החיבור מעדכנת דגל בשם Z ("דגל האפס") באוגר הסטטוס (PSW).	cmp A, r1	אם תוכן המשתנה A זהה לתוכנו של אוגר r1 אז דגל האפס, Z, באוגר הסטטוס (PSW) יודלק, אחרת הדגל יאפס.
add	אופrnd היעד (השני) מקבל את תוצאה החיבור של תוכן המשתנה (הראשון) והיעד (השני).	add A, r0	אוגר 0 מקבל את תוצאה החיבור של תוכן המשתנה A ותוכנו הנוכחי של r0.

הסבר הדוגמה	דוגמה	הסבר הפעולה	הוראה
אוגר $r_1$ מקבל את תוצאה החישור של הערך 3 מתוכנו הנוכחי של האוגר $r_1$ .	sub #3, r1	אופרנד היעד (השני) מקבל את תוצאה החישור של אופרנד המקור (הראשון) מאופרנד היעד (השני).	sub
המען שמייצגת התוויות HELLO מוצב לאוגר $r_1$ .	lea HELLO, r1	lea הוא קיצור (ראשי תיבות) של lea effective address מציבה את המعن בזיכרון המיוצג על ידי התוויות שבאופרנד הראשון (המקור), אל אופרנד היעד (האופרנד השני).	lea

#### קבוצת ההוראות השניה:

אליהן ההוראות הדורשות אופרנד אחד בלבד. אופן הקידוד של האופרנד הוא כמוו של אופרנד היעד בפקודה עם שני אופרנדים. במקרה זה, השדה של אופרנד המקור (סיביות 4-5) בambilת הראשונה בקידוד ההוראה הינו חסר משמעות, ולפיכך יכול 00.

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

הסבר הדוגמה	דוגמה	הסבר הפעולה	הוראה
$r_2 \leftarrow \text{not } r_2$	not r2	היפוך ערכי הסיביות באופרנד (כל סיבית שערכה 0 הפוך ל-1 ולהיפך ל-0).	Not
$r_2 \leftarrow 0$	clr r2	איפוס תוכן האופרנד.	clr
$r_2 \leftarrow r_2 + 1$	inc r2	הגדלת תוכן האופרנד באחד.	inc
$C \leftarrow C - 1$	dec C	הקטנת תוכן האופרנד באחד.	dec
PC $\leftarrow$ LINE מצביע התוכנית מקבל את המعن המיוצג על ידי התוויות LINE, ולפיכך הפקודה הבאה שתבוצע תהיה בمعنى זה.	jmp LINE	קפיצה (הסתעפות) בלתי מותנית אל ההוראה שנמצאת בمعنى המיוצג על ידי האופרנד. ככלומר, בעת ביצוע ההוראה, מצביע התוכנית (PC) קיבל את ערך אופרנד היעד אם ערכו של הדגל Z באוגר הסטטוס (PSW) הינו 0. כזכור, הדגל Z נקבע בפקודת cmp.	jmp
אם ערך הדגל Z באוגר הסטטוס (PSW) הינו 0, אז: $PC \leftarrow \text{LINE}$	bne LINE	bne הוא קיצור (ראשי תיבות) של branch if not equal (to zero). זוהי הוראות הסתעפות מותנית. מצביע התוכנית (PC) קיבל את ערך אופרנד היעד אם ערכו של הדגל Z באוגר הסטטוס (PSW) הינו 0. כזכור, הדגל Z נקבע בפקודת cmp.	bne
קוד ה-ascii של התו הנקרא מהקלט ייבנס לאוגר $r_1$ .	red r1	קריאה שלתו מהקלט הסטנדרטי (stdin) אל האופרנד.	red
התו אשר קוד ה-ascii שלו נמצא באוגר $r_1$ יודפס לפלט הסטנדרטי.	prn r1	הדפסת התו הנמצא באופרנד, אל הפלט הסטנדרטי (stdout).	prn
push(PC) $PC \leftarrow \text{FUNC}$	jsr FUNC	קריאה לשגרה (סברוטינה), מצביע התוכנית (PC) הנוכחי נדחף לתוך המחסנית שבזיכרון המחשב, והאופרנד מוכנס ל-PC.	Jsr

### **קובצת ההוראות השלישי:**

אלו הן הוראות ללא אופרנדים. קידוד ההוראה מורכב ממילה אחת בלבד. במקרה זה, השודות של אופרנד המקור ואופרנד היעד אינם רלוונטיים (כי אין אופרנדים), ויכילו 0.

ההוראות השויות לקובצת זו הן : rts, stop .

הוראה	עיצירת ריצת התוכנית.	דוגמה	הסבר הדוגמה
Rts		rts	חרזה משיגרה. הערך שנמצא בראש המחסנית של המחשב מוצא מן המחסנית, ומוכנס אל מצביע התוכנית (PC).
Stop		stop	התוכנית עוצרת.

### מבנה שפת האסמבלי:

שפת האסמבלי בנויה ממשפטים (statements). קובץ בשפת אסמבלי מורכב משורות המכילות ממשפטים של השפה, כאשר כל ממשפט מופיע בשורה נפרדת. ככלומר התו המפריד בין ממשפט למשפט בקובץ הינו התו 't' (שורה חדשה).

אורכה של שורה בקובץ המקור הוא 80 תווים לכל היותר (לא כולל התו t).

ישנם חמישה סוגים ממשפטים (שורות) בשפת האסמבלי, והם :

סוג המשפט	הסבר כללי
משפט ריק	זהו שורה המכילה אך ורק תווים לבנים (whitespace), ככלומר רק את התווים ' ' ו- 't' (רווחים וטאבים). ייתכן ובשורה אין אף תו (למעט התו t), ככלומר השורה ריקה.
משפט הערת	זהו שורה בה התו הראשון הינו ';' (נקודה פסיק). על האסמבלי להתעלם לחולותן משורה זו.
משפט הנטהיה	זהו משפט המנחה את האסמבלי מה עליו לעשות כשהוא פועל על תוכנית המקור. יש מספר סוגים של משפטי הנטהיה.משפט הנטהיה עשוי לגורום להקצת זיכרון ואתחול משתנים של התוכנית, אך הוא אינו מייצר קידוד של הוראות מכונה המיועדות לביצוע בעת ריצת התוכנית.
משפט הוראה	זהו משפט המייצר קידוד של הוראות מכונה לביצוע בעת ריצת התוכנית. המשפט מורכב ממשם של הוראה שעל המעבד לבצע, ותיאור האופרנדים של הוראה.
משפט מאקרו	זהו משפט באמצעותו ניתן להגדיר שם סימבולי המייצג קבוע מספרי. במהלך הפעלה, בכל מקום בו מופיע השם, הוא יוחלף בקבוע המספרי. משפט זה לכשעצמו אינו מייצר קוד ואינו מקצה זיכרון.

כעת נפרט יותר לגבי סוגי המשפטים השונים.

### משפט הנטהיה:

משפט הנטהיה הוא בעל המבנה הבא :

בתחילת המשפט יכולה להופיע תוית (label). התוית חייב להיות בתחביר חוקי (התחביר של תוית חוקית יתואר בהמשך). התוית היא אופציונלית.

לאחר מכן מופיע שם הנטהיה. לאחר שם הנטהיה יופיעו פרמטרים (מספר הפרמטרים בהתאם להנטהיה). שם של הנטהיה מתחילה בתו ';' (נקודה) ולאחריו תווים באותיות קטנות (lower case) בלבד.

יש לשים לב: **למיילים בקוד המכונה הנוצרות משפט הנטהיה לא מצורף השדה E,R,A, והקידוד ממלא את כל 14 היסיות של המילה.**

יש ארבעה סוגים (שמות) של משפטי הינה, והם :

## 1. הינה 'data'.

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

.data 7, -57, +17, 9

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

המשפט 'data' מנהה את האSEMBLER להקצות מקום בתמונת הנתונים (image), אשר בו יອחسن הערכים של הפרמטרים, ולאחר מכן מונה הנתונים, בהתאם למספר הערכים. אם בהנחתה data. מוגדרת תווית, אז תווית זו מקבלת את ערך מונה הנתונים (לפניהם הקידום), ומוכנסת אל טבלת הסמלים. דבר זה מאפשר להתייחס אל מקום מסוים בתמונת הנתונים דרך שם התווית (למעשה, זהה דרך להגדיר שם של משתנה).

כלומר אם נכתב :

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

אז יוקצו בתמונת הנתונים ארבע מילימ רצופות שיכילו את המספרים שמופיעים בהינה. התווית XYZ מזוהה עם כתובות המילה הראשונה.

אם נכתב בתכנית הוראת מכונה :

mov XYZ, r1

אז בזמן ריצת התכנית יוכנס לאוגר r1 הערך 7.

ואילו הוראה :

lea XYZ, r1

תכנס לאוגר r1 את ערך התווית XYZ (כלומר הכתובת בזיכרון בה מאוחSEN הערך 7).

## 2. הינה 'string'.

הינה 'string' פרט אחד, שהוא מחרוזת חוקית. תווים המחרוזת מקודדים לפי ערכי ה-ascii המתאים, ומוכנסים אל תמיון הנתונים לפי סדרם, כל TWO במילה נפרדת. בסוף המחרוזת יתווסף התו '0' (הערך המספרי 0), המסמן את סוף המחרוזת. מונה הנתונים של האSEMBLER יקודם בהתאם לערך המחרוזת (בתוספת מקום אחד עבור התו המסיים). אם בשורת ההינה מוגדרת תווית, אז תווית זו מקבלת את ערך מונה הנתונים (לפניהם הקידום) ומוכנסת אל טבלת הסמלים, בדומה כפי שנעשה עבור 'data'. (כלומר ערך התווית יהיה הכתובת בזיכרון שבה מתחילה המחרוזת).

לדוגמה, הינה :

STR: .string "abcdef"

מקצת בתמונת הנתונים רצף של 7 מילימ, ומאחלה את המילימ לקובי ascii של התווים לפי הסדר במחרוזת, ולאחריהם הערך 0 לסימון סוף מחרוזת. התווית STR מזוהה עם כתובות התחלה המחרוזת.

### 3. הנקה 'entry'.

לנקה 'entry', פרמטר אחד, והוא שם של תוית המוגדרת בקובץ המקור הנוכחי (כלומר תוית שמקבלת את ערכיה בקובץ זה). מטרת הנקה entry. היא לאפיין את התוית הזה באופן שיאפשר לקוד אסמליל הנמצא בקבצי מקור אחרים להשתמש בה (אופrnd של הורה).

לדוגמה, השורות:

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

מודיעות לאסמליל שאפשר להתיחס מקובץ אחר לטוית O HELLO המוגדרת בקובץ הנוכחי.

لتשומת לב: תוית המוגדרת בתחלת שורת entry. הינה חסרת משמעות והאסמליל **מתעלם** מתוויות זו (אפשר שהאסמליל יוציא הודעה אחרת).

### 4. הנקה 'extern'.

לנקה 'extern', פרמטר אחד, והוא שם של תוית שאינה מוגדרת בקובץ המקור הנוכחי. מטרת ההורה היא להודיע לאסמליל כי התוית מוגדרת בקובץ מקור אחר, וכי קוד אסמליל בקובץ הנוכחי עושה בתוית שימוש.

שים לב כי הנקה זו תואמת להנקה 'entry', המופיעה בקובץ בו מוגדרת התוית. בשלב הקישור התבכע התאמה בין ערך התוית, כפי שנקבע בקוד המכונה של הקובץ שהגדיר את התוית, לבין קידוד ההוראות המשמשות בתוית בקבצים אחרים (שלב הקישור אינו רלוונטי לממ"ז זה).

לדוגמה, משפט הנקה 'extern', התואם למשפט הנקה 'entry' מהדוגמה הקודמת יהיה:

.extern	HELLO
---------	-------

لتשומת לב: תוית המוגדרת בתחלת שורת extern. הינה חסרת משמעות והאסמליל **מתעלם** מתוויות זו (אפשר שהאסמליל יוציא הודעה אחרת).

**משפט הורה:**

משפט הורה מורכב מחלקים הבאים, לפי הסדר:

1. תוית אופציונלית.
2. שם הפעולה.
3. אופrndים בהתאם לסוג הפעולה (בין 0 ל-2 אופrndים).

אם מוגדרת תוית בshort הורה, אז היא תוכנס אל טבלת הסמלים. ערך התוית יהיה מען המילה הראשונה של הורה בתחום הקוד שבונה האסמליל.

שם הפעולה תמיד באותיות קטנות (lower case), והוא אחת מ- 16 הפעולות שפורטו לעיל.

לאחר שם הפעולה יכולים להופיע אופrndים (אחד או שניים), בהתאם לסוג הפעולה.

כאשר יש שני אופrndים, האופrndים מופרדים זה מזה ב' ', (פסיק). בדומה להנקה 'data', לא חייבת להיות הצמדה של האופrndים לפסיק או לשם הפעולה באופן כלשהו. כל כמות של רווחים ו/או טאים בין האופrndים לפסיק, או בין שם הפעולה לאופrnd הראשון, היא חוקית.

משפט הורה עם שני אופrndים המבנה הבא:

<u>תוית-אופציונלית:</u>	<u>שם-הפעולה</u>
-------------------------	------------------

<u>אופrnd-יעד, אופrnd-מקור</u>
--------------------------------

לדוגמה:

HELLO:	add r7, B
--------	-----------

למשפט הוראה עם אופרנד אחד המבנה הבא :

תווית-אופציאונלית שם-הפעולה אופרנד

לדוגמה : HELLO: bne XYZ

למשפט הוראה ללא אופרנדים המבנה הבא :

תווית-אופציאונלית שם-הפעולה

לדוגמה : END: stop

### **משפט מאקרו:**

משפט מאקרו הוא בעל המבנה הבא :

.קבוע-מספרי = שם-המאקרו .define

לדוגמה :

.define len = 4  
.define init = -3

הרעיון הוא ליצג קבוע מספרי באמצעות שם סימבולי. בכל מקום בתוכנית בו מופיע שם של מאקרו, האסמלר יחליף בקידוד הפקודה לקוד מכוונה את השם קבוע המספריא אליו הוגדר.

המילה השמורה 'define', נוטנה באותיות קטנות בלבד. התחברר של שם המacro זהה לת לחבר של תווית. אסור להגדיר את אותו שם מאקרו יותר מפעם אחת. כמו כן אותו סמל לא יכול לשמש חן כשם של מאקרו והן כתווית באורה תכנית. מילים שמורות של שפת האסמלר (שם של אוגר, שם של הוראת מכוונה או שם של הנחיה) אין יכולות לשמש שם של מאקרו.  
הקבוע המספריא הוא שלם בסיס עשרוני.  
בין שם המacro לקבוע מפ прид התו '='. מותרים תווים לבנים בשני צידי התו.

macro חייב להיות מוגדר לפני השימוש הראשוני בו.  
אסור להגדיר תווית בשורה שהיא המשפט מאקרו.

ניתן להשתמש בשם המacro בכל מקום בתוכנית האסמלר בו יכול להופיע קבוע מספרי, כמובן : **איןדקס בשיטת מיון אינדקס ישיר, או ערך בשיטת מיון מיידי, או אופרנד של הנחית data.**

לדוגמה, בהינתן הגדרות המacro לעיל, אזי ההוראה :

mov x[len], r3

תעתיק את האיבר באינדקס 4 במערך x אל אוגר r3.

וההוראה :

mov #init, r2

תציב את הערך המיידי 3 - אל האוגר r2

כמו כן, ההנחיה :

.data len

תקצה מילה בזיכרון עם ערך התחלתי 4.

## אפיון השדות במשפטים של שפט האסמבלי

תווית:

תווית חוקית מתחילה באות אלפביתית (גדולה או קטנה), ולאחריה סדרה כלשהי של אותיות אלפביתיות (גדולות או קטנות) ו/או ספרות. האורך המקסימלי של תווית הוא 31 תווים.

**הגדרה של תווית מסתויימת בתו '** (נקודותים).תו זה אינו מהו חלק מהתווית, אלא רק סימן המציין את סוף ההגדרה.

אסור שאונה תווית תוגדר יותר מפעם אחת (כਮון בשורות שונות).אותיות קטנות וגדולות נחשבות שונות זו מזו. כמו כן, אסור שאותו סמל ישמש הן כתווית והן שם של מאקרו.

לדוגמא, התוויות המוגדרות להלן הן תוויות חוקיות.

hEllo:

x:

He78902:

**لتשומת לב:** מיללים שמורות של שפט האסמבלי (כלומר שם של פעולה או החלטה, או שם של אוגר) אינן יכולות לשמש גם שם של תווית.

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

מספר:

מספר חוקי מתחילה בסימן אופציוני: ‘+’, או ‘-’, ולאחריו סדרה כלשהי של ספרות בסיס עשרוני. לדוגמה: 5,-, 76, +123 הם מספרים חוקיים. אין תמיכה בשפט אסמבלי ביצוג בסיס אחר מאשר עשרוני, ואין תמיכה במספרים שאינם שלמים.

מחuzeות:

מחuzeות חוקית היא סדרת תווים `ascii` נראים (שניתנים להדפסה), המוקפים במרקאות כפולות (המרקאות אינן נחשבות חלק מהמחuzeות). דוגמה למחuzeות חוקית: “hello world”.

## תפקיד השדה E,R,A בקוד המכוונה

בכל מילה בקוד המכוונה של הוראה (לא של נתוניים), האסמבלי מכניס מידע עבור תהליך הקישור והטיענה. זהו השדה A,R,E (שתי הסיביות הימניות 0-1). המידע ישמש לתיקונים בקוד בכל פעם שייטען לזרקן לזרק הרצה. האסמבלי בונה מלכתחילה קוד שמיועד לטיענה החל מכתובת 100. התיקונים יאפשרו לטיען את הקוד בכל פעם למקום אחר, בלי צורך לחזור על תהליך האסמבלי.

שתי הסיביות בשדה E,A,R יכilio את אחד הערכים הבינאים: 00, 01, או 10. המשמעות של כל ערך מפורטת להלן.

האות ‘A’ (קייזר של absolute) בא להציג שתוכן המילה אינו תלוי במקום בו זיכרו בו יייטען בפועל קוד המכוונה של התכנית בעת ביצועה (למשל מילה המכילה אופrnd מדוי). במקרה זה שתי הסיביות הימניות יכilio את הערך 00.

האות ‘R’ (קייזר של relocatable) בא להציג שתוכן המילה תלוי במקומות בו זיכרו בו יייטען בפועל קוד המכוונה של התכנית בעת ביצועה (למשל מילה המכילה כתובת של תווית המוגדרת בקובץ המקור). במקרה זה שתי הסיביות הימניות יכilio את הערך 10.

האות 'E' (קייזור של [external]) בא להציג שטוכן המילה תלוי בערכו של סמל חיצוני (external). (למשל מילה המכילה כתובת של תווית חיצונית, כלומר תווית שאינה מוגדרת בקובץ המקור הנוכחי). במקרה זה שתי הסיבות הימניות יכilo את הערך 01.

### אסמבילר עם שני מעבריים

כאשר מקבל האסמבילר קוד בשפת אסמבלי לתרגם, עליו לבצע שתי משימות עיקריות לצורך הכתנת הקוד הבינארי: הראשונה היא לזהות ולתרגם את שמות הפעולות, והשנייה היא לקבוע מענים לכל הסמלים המופיעים בתוכנית.

לדוגמה: האסמבילר מקבל את הקוד הבא:

```
.define sz = 2
MAIN:      mov   r3, LIST[sz]
LOOP:       jmp   L1
            prn   #-5
            mov   STR[5], STR[2]
            sub   r1, r4
            cmp   r3, #sz
            bne   END
L1:        inc   K
            bne   LOOP
END:       stop

.define len = 4
STR:        .string "abcdef"
LIST:       .data  6, -9, len
K:          .data  22
```

עליו להחליף את שמות הפעולות stop בקוד הבינארי  
הSKUול להם במודל המחשב שהגדנו.

כמו כן, על האסמבילר להחליף את הסמלים K,STR, LIST, L1, MAIN, LOOP, END במשמעותם של המיקומות בזיכרון שם נמצאים כל נתון או הוראה בהתאם.

בנוסף, על האסמבילר להחליף את שמות המאקרוים len ו- sz בקבועים המספריים שהם ערכי המאקרוים בהתאם, בכל מקום בו הם מופיעים.

אנו מניחים שקוד המconaה של התכנית (הוראות ונתונים) ייבנה כך שיתאים לטעינה בזיכרון החל ממען 100 (בבסיס עשרוני). מתќבל התרגום הבא לקוד מכונה בינהי:

Decimal Address	Source Code	Explanation	Binary Machine Code
0100	MAIN: mov r3, LIST[sz]	First word of instruction	00000000111000
0101		Source register 3	00000001100000
0102		Address of label LIST (integer array)	00001000010010
0103		Value of macro sz (index 2)	000000000001000
0104	LOOP: jmp L1		00001001000100
0105		Address of label L1	00000111100010
0106	prn #-5		000011000000000
0107		Immediate value -5	1111111101100
0108	mov STR[5], STR[2]		00000000101000
0109		Address of label STR (string)	00000111110010
0110		Index 5	000000000010100
0111		Address of label STR	00000111110010
0112		Index 2	000000000001000

Decimal Address	Source Code	Explanation	Binary Machine Code
0113	sub r1, r4		00000011111100
0114		Source register 1 and target register 4	00000000110000
0115	cmp r3, #sz		00000001110000
0116		Source register 3	00000001100000
0117		Value of macro sz (immediate #2)	00000000001000
0118	bne END		00001010000100
0119		Address of label END	00000111110010
0120	L1: inc K		00000111000100
0121		Address of label K (integer)	00001000011110
0122	bne LOOP		00001010000100
0123		Address of label LOOP	00000110100010
0124	END: stop		00001111000000
0125	STR: .string "abcdef"	Ascii code 'a'	00000001100001
0126		Ascii code 'b'	00000001100010
0127		Ascii code 'c'	00000001100011
0128		Ascii code 'd'	00000001100100
0129		Ascii code 'e'	00000001100101
0130		Ascii code 'f'	00000001100110
0131		Ascii code '\0' (end of string)	0000000000000000
0132	LIST: .data 6, -9, len	Integer 6 (first in array of 3 words) Integer -9 Value of macro len (integer 4)	000000000000110 11111111110111 000000000000100
0133			
0134			
0135	K: .data 22	Integer 22 (single word)	00000000010110

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

כדי לעשות המרה לבינארי של אופרנדים שהם מענים סמליים (תוויות), יש צורך לבנות טבלה דומה. אולם בהבדל מהקודים של הפעולות, הידיעים מראש, הרו המעניינים בזיכרון עבור הסמלים בשימוש התכנית אינם ידועים, עד אשר תוכנית המקור נסקרה כולה ונתגלו כל הגדרות הסמלים.

למשל, בקוד לעיל, האסמבלר אינו יכול לדעת שהסמל END אמור להיות משוייך למן 124 (עשרוני), והסמל K אמור להיות משוייך למן 135, אלא רק לאחר שנקרוו כל שורות התכנית.

לכן מפרידים את הטיפול של האסמבלר בסמלים לשני שלבים. בשלב הראשון בונים טבלה של כל הסמלים, עם הערכים המספריים המשוכחים 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 משוייך ערך מסווני: שהוא מין בזיכרון או ערך קבוע של מקאו. בדוגמה לעיל, טבלת הסמלים לאחר מעבר ראשוני היא:

סמל	ערך (בסיס עשרוני)
sz	2
MAIN	100
LOOP	104
L1	120
END	124
len	4
STR	125
LIST	132
K	135

במעבר השני נעשית ההמרה של קוד המקור לקוד מוכנה. בתחילת המעבר השני צרכיים הערכיים של הסמלים להיות כבר ידועים.

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

### המעבר הראשון

במעבר הראשון נדרשים כלים כדי לקבוע איזה מען ישוויך לכל סמל. העיקרונו הבסיסי הוא לספק את המוקומות בזיכרונו, אותן תופסות ההוראות. אם כל הוראה תיטען בזיכרונו במקום העוקב להוראה הקודמת, תציג ספירה כזאת את מען ההוראה הבאה. הספירה נעשית על ידי האסטブル ומוחזקת במבנה ההוראות (IC). ערכו ההתחלתי של IC הוא 100 (עשורי), ולכן קוד המוכנה של ההוראה הראשונה נבנה כך שייטען לזכרונו החל ממען 100. ה-IC מתעדכן בכל שורת הוראה המקצת מקום בזיכרונו. לאחר שהאסטブル קובע מהו אורך ההוראה, ה-IC מוגדל במספר התאים (מיילים) הנתפסים על ידי ההוראה, וכך הוא מציבע על התא הפוי הבא.

כאמור, כדי לקודד את ההוראות בשפת מוכנה, מחזיק האסטブル טבלה, שיש בה קוד מתאים לכל שם פעולה. בזמן התרגום מחליף האסטブル כל שם פעולה בקוד שלה, וכל אופרנד בקידוד מתאים. אך פעולה ה恰恰פה אינה כה פשוטה. ההוראות משתמשות בשיטות מייען מגוונות לאופרנדים. אותה פעולה יכולה לקבל משמעותות שונות, בכל אחת משיטות המייען, ולכן ניתן למגוון קידודים שונים לפי שיטות המייען. לדוגמה, פעולה ההזזה *smt* יכולה להתיחס להעתיקת תוכן תא זיכרון לאוגר, או להעתיקת תוכן אוגר לאוגר אחר, וכן הלאה. לכל אפשרות כזו של *smt* עשוי להתאים קידוד שונה.

על האסטブル לסרוק את שורת ההוראה בשלמותה, ולהחליט לגבי הקידוד לפי האופרנדים. בדרך כלל מחלק הקידוד לשדה של שם הפעולה, ושדות נוספים המכילים מידע לגבי שיטות המייען.

במחשב שלנו קיימת גמישות לגבי שיטות המייען של כל אחד מהאופרנדים בנפרד. **הערה:** דבר זה לא מחייב לגבי כל מחשב. ישנים מחשבים בהם, למשל, כל הפקודות הן בעלות אופרנד יחיד (והפעולות מתבצעות על אופרנד זה ועל אוגר קבוע). יש גם מחשבים עם פקודות של שלשה אופרנדים (כasher האופרנד השלישי משמש לאחסון תוצאת הפעולה), ועוד אפשרויות אחרות.

כאשר נתקל האסטブル בתווית המופיעה בתחילת השורה, הוא יודע שלפניו הגדרה של תווית, והוא משייך לה מען – תוכנו הנוכחי של 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 האסטブル לשולף את המען המתאים מטבלת הסמלים.

הוראה יכולה להתיחס גם לסמל שטרם הוגדר עד כה בתכנית, אלא יוגדר רק בהמשך התכנית. להלן דוגמה, הוראות הסתעפות למנע שימוש אוגרי על ידי התווית A שמוינעת רק בהמשך הקוד :

A:	bne      A .....
	• • •

כאשר מגיע האסטブル לשורת הסתעפות (bne A), הוא טרם נתקל בהגדרת התווית A ומוגן לא נתן לה מען, ולכן איןו יכול להחליף את הסמל A (האופרנד של ההוראה smt) במשמעותו בזיכרונו. נראה בהמשך כיצד נפתרת בעיה זו.

בכל מקרה, תמיד אפשר לבנות מעבר הראשון את הקוד הבינארי המלא של המילה הראשונה של כל הוראה, וכן את הקוד הבינארי של כל הנתונים (המתקבלים מהחניות .string, .data ..).

## המעבר השני

ראינו שבמעבר הראשון, האסטבלר אינו יכול לבנות את קוד המוכנה של אופרנדים המשמשים בסמלים שעדיין לא הוגדרו. רק לאחר שהאסטבלר עבר על כל התכנית, כך שכל הסמלים נכנסו כבר לטבלת הסמלים, יוכל האסטבלר להשלים את קוד המוכנה של כל האופרנדים.

לשם כך עובר האסטבלר שנייה על כל קובץ המקור, וمعدכן את קוד המוכנה של האופרנדים המשמשים בסמלים, באמצעות ערכיו הקיימים בטבלת הסמלים.  
זהו המעבר השני, ובסיומו תהיה התוכנית מתרגמת בשמותה לקוד מכונה.

### הפרדת הוראות ונתונים

בתכנית מבחןים בשני סוגים של תוכן: הוראות ונתונים. יש לארגן את קוד המוכנה כך שתהייה הפרדה בין הנתונים וההוראות. הפרדת ההוראות והנתונים לקטעים שונים בזיכרון היא שיטה עדיפה על פני הצמדה של הגדרות הנתונים להוראות המשמשות בהן.

אחת הסכנות הטמונה באירוע ההוראות מהנתונים היא, שלפעמים עלול המעבד, בעקבות שגיאה לוגית בתכנית, לנסוטו "לבצע" את הנתונים כאילו היו הוראות חוקיות. למשל, שגיאה שיכולה לגרום לתופעה כזו הסתעפות לא נכונה. התכנית כMOV לא תעבור נכוון, אך לרוב הנזק הוא יותר חמור, כי נוצרת חריגת חומרה ברגע שהמעבד מבצע פעולה שאינה חוקית.

האסטבלר שלנו חייב להפריד, בקוד המוכנה שהוא מיצר, בין קטע הנתונים לקטע ההוראות. כלומר בקובץ הפלט (בקוד המכונה) תהיה הפרדה של הוראות ונתונים לשני קטעים נפרדים, ואילו בקובץ הקלט אין חובה שתהייה הפרדה כזו. בהמשך מתואר אלגוריתם של האסטבלר, ובו פרטים כיצד לבצע את הפרדה.

### גילוי שגיאות בתכנית המקור

האסטבלר אמר לגלות ולדוח על שגיאות בתחביר של תכנית המקור, כגון פעולה שאינה קיימת, מספר אופרנדים שגוי, סוג אופרנד שאינו מתאים לפעולה, שם אוגר לא קיים, ועוד שגיאות אחרות. כמו כן מודיא האסטבלר שככל סמל מוגדר פעמי אחת בדיקות.

מכאן, שככל שגיאה המתגללה על ידי האסטבלר נגרמת (בדרך כלל) על ידי שורת קלט מסויימת. לדוגמה, אם מופיעים שני אופרנדים בהוראה שאמור להיות בה רק אופרנד יחיד, האסטבלר ייתן הודעת שגיאה בנוסח "יותר מדי אופרנדים".

על כל הודעת שגיאה לציין גם את מספר השורה בתכנית המקור בה זוהתה השגיאה.

לתשומתך: האסטבלר אינו עוצר את פועלתו אחרי שנמצאה השגיאה הראשונה, אלא ממשיך לעبور על הפלט כדי לגלוות שגיאות נוספות, ככל שישן. כMOV שאין כל טעם לייצר את קבצי הפלט אם נתגלו שגיאות (ممילא אי אפשר להשלים את קוד המוכנה).

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

שם פעולה	שיטות מייען חוקיות עבור אופרנד מקור	שיטות מייען חוקיות עבור אופרנד יעד
mov	0,1,2,3	1,2,3
cmp	0,1,2,3	0,1,2,3
add	0,1,2,3	1,2,3
sub	0,1,2,3	1,2,3
not	אין אופרנד מקור	1,2,3
clr	אין אופרנד מקור	1,2,3
lea	1,2	1,2,3
inc	אין אופרנד מקור	1,2,3

שיטות מייען חוקיות עבור אופרנד יעד	שיטות מייען חוקיות עבור אופרנדמקור	שם פעולה
1,2,3	אין אופרנד מקור	dec
1,3	אין אופרנד מקור	jmp
1,3	אין אופרנד מקור	bne
1,2,3	אין אופרנד מקור	red
0,1,2,3	אין אופרנד מקור	prn
1,3	אין אופרנד מקור	jsr
אין אופרנד יעד	אין אופרנד מקור	rts
אין אופרנד יעד	אין אופרנד מקור	stop

### אלגוריתם שלדי של האסטבלר

לחידוד ההבנה של תהליך העבודה של האסטבלר, נציג להלן אלגוריתם שלדי למעבר הראשון ולמעבר השני. לתשומתך: אין חובה להשתמש דווקא באלגוריתם זה.

אנו מחלקים את קוד המכונה לשני אזוריים, אזור ההוראות (code) ואזור הנתונים (data). לכל אזור יש מונה משלה, ונסמנם IC (МОНЯ ХОРОАТ - Instruction-Counter) ו-DC (МОНЯ НАТОНИМ - Data-Counter). **נבנה את קוד המכונה כך שתיאים לטעינה זיכרון החל מכתובת 100.**

כמו כן, נסמן ב- L את מספר המיללים שתופס קוד המכונה של הוראה נתונה.

בכל מעבר מתחילה לקרוא את קובץ המקור מהתחלתה.

#### מעבר ראשון

1. אתחל 0  $IC \leftarrow 0$ ,  $DC \leftarrow 1$ .
2. קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עברו ל-16.
3. האם זהה הגדרת מקאו? אם לא, עברו ל-5.
4. הכנס את שם המאקרו לטבלת הסמלים עם המאפיין macro. ערכו יהיה כפי שמופיע בהגדרת. (אם הסמל כבר נמצא בטבלה, יש להודיע על שגיאיה). חזרו ל-2.
5. האם השדה הראשון בשורה הוא סמל? אם לא, עברו ל-7.
6. הדלק דגל "יש הגדרת סמל".
7. האם זהה הначיה לאחסון נתונים, ככלומר, האם הначיה data.string? אם לא, עברו ל-10.
8. אם יש הגדרת סמל (תוויות), הכנס אותו לטבלת הסמלים עם המאפיין DC. ערכו יהיה DC.
9. זהה את סוג הנתונים, קודד אותם בזיכרון, ועדכן את מונה הנתונים DC בהתאם לאורכם. אם זהה הначיה data., ויש בה נתון שהוא סמל, בדוק שהסמל מופיע בטבלה עם המאפיין macro, והשתמש בערכו. (אם הסמל אינו בטבלה או לא מאופיין כ- macro, יש להודיע על שגיאיה). חזרו ל-2.
10. האם זו הначיה extern. או הначיה entry.? אם לא, עברו ל-12.
11. האם זהה הначיה extern.? אם כן, הכנס כל סמל (אחד או יותר) המופיע כאופרנד של הначיה לתוך טבלת הסמלים ללא ערך, עם המאפיין external. חזרו ל-2.
12. אם יש הגדרת סמל, הכנס אותו לטבלת הסמלים עם המאפיין code. ערכו יהיה IC+100 (אם הסמל כבר נמצא בטבלה יש להודיע על שגיאיה).
13. חפש את שם הפעולה בטבלת שמות הפעולות, ואם לא נמצא – הודיע על שגיאיה בשם הוראה.
14. נתח את מבנה האופרנדים של ההוראה וחשב את L. בנה CUT את הקוד הבינארי של המילה הראשונה של ההוראה.
15. עדכן  $L \leftarrow IC + 1$ , וחזרו ל-2.
16. קובץ המקור נקרא בשלמותו. אם נמצאו שגיאות במעבר הראשון, עצור כאן.
17. עדכן בטבלת הסמלים את ערכו של כל סמל המופיע כ- data , ע"י הוספת הערך IC+100 (ראה הסבר בהמשך).
18. התחל מעבר שני.

## מעבר שני

- .1. אתחל 0  $IC \leftarrow 0$
- .2. קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עבור ל- 9.
- .3. אם השדה הראשון הוא סמל, דרג עליו.
- .4. האם זהה הנחיתת.data או .extern ? אם כן, חזור ל-2.
- .5. האם זהה הנחיתת.entry ? אם לא, עבור ל-7.
- .6. סמן בטבלת הסמלים את הסמלים המתאימים במאפיין entry. חזור ל-2.
- .7. השלם את קידוד האופרנדים החל מהמילה השנייה בקוד הבנאי של ההוראה, בהתאם לשיטת המיעון. אם אופרנד מכיל סמל, מצא את הערך בטבלת הסמלים (אם הסמל לא נמצא בטבלה, יש להודיע על שגיאה).
- .8. עדכן  $L \leftarrow IC + L$ , וחרור ל-2.
- .9. קובץ המקור נקרא בשלמותו. אם נמצא שגיאות במהלך השני, עוזר כאן.
- .10. בנה את קבצי הפלט (פרטים נוספים בהמשך).

נפעיל אלגוריתם זה על תוכנית הדוגמה שהציגנו קודם :

```
.define sz = 2
MAIN:    mov   r3, LIST[sz]
LOOP:     jmp   L1
          prn   #-5
          mov   STR[5], STR[2]
          sub   r1, r4
          cmp   r3, #sz
          bne   END
L1:       inc   K
          bne   LOOP
END:      stop
.define len = 4
STR:      .string "abcdef"
LIST:    .data  6, -9, len
K:        .data  22
```

בוצע עתה מעבר ראשון על הקוד הנוכחי. בונה את טבלת הסמלים. כמו כן, נבצע במעבר זה גם את קידוד כל הנתונים, וקידוד של המילה הראשונה של כל הוראה. כמו כן ניתן לקודד מילים נוספות של כל ההוראה, כל עוד קידוד זה אינו תלוי בכתובות של תוויות. את החלקים שעדיין לא מקודדים במעבר זה, נשאיר כמותם שהם (מסומנים ב- ? בדוגמה להלן).

שים לב שקוד המכונה בונה לטעינה לזיכרון החל מהמען 100 (בבסיס עשרוני).

Decimal Address	Source Code	Explanation	Binary Machine Code
0100	MAIN: mov r3, LIST[sz]	First word of instruction	00000000111000
0101		Source register 3	00000001100000
0102		Address of label LIST (integer array)	?
0103		Value of macro sz (index 2)	00000000001000
0104	LOOP: jmp L1		00001001000100
0105		Address of label L1	?
0106	prn #-5		00001100000000
0107		Immediate value -5	1111111101100
0108	mov STR[5], STR[2]		00000000101000
0109		Address of label STR (string)	?
0110		Index 5	00000000010100
0111		Address of label STR	?
0112		Index 2	00000000001000

Decimal Address	Source Code	Explanation	Binary Machine Code
0113 0114	sub r1, r4	Source register 1 and target register 4	00000011111100 00000000110000
0115 0116 0117	cmp r3, #sz	Source register 3 Value of macro sz (immediate #2)	00000001110000 00000001100000 00000000001000
0118 0119	bne END	Address of label END	00001010000100 ?
0120 0121	L1: inc K	Address of label K (integer)	00000111000100 ?
0122 0123	bne LOOP	Address of label LOOP	00001010000100 ?
0124	END: stop		00001111000000
0125	STR: .string "abcdef"	Ascii code 'a'	00000001100001
0126		Ascii code 'b'	00000001100010
0127		Ascii code 'c'	00000001100011
0128		Ascii code 'd'	00000001100100
0129		Ascii code 'e'	00000001100101
0130		Ascii code 'f'	00000001100110
0131		Ascii code '\0' (end of string)	00000000000000
0132 0133 0134	LIST: .data 6, -9, len	Integer 6 (first in array of 3 words) Integer -9 Value of macro len (integer 4)	000000000000110 1111111110111 000000000000100
0135	K: .data 22	Integer 22 (single word)	00000000010110

טבלת הסמלים :

סמל	מופיעים	ערך (בסיס עשרוני)
sz	macro	2
MAIN	code	100
LOOP	code	104
L1	code	120
END	code	124
len	macro	4
STR	data	125
LIST	data	132
K	data	135

נבע עתה את המעבר השני. נשלים את הקידוד החסר באמצעות טבלת הסמלים, ונרשום את הקוד בצורתו הסופית :

Decimal Address	Source Code	Binary Machine Code
0100 0101 0102 0103	MAIN: mov r3, LIST[sz]	00000000111000 00000001100000 0001000010010 000000000001000
0104 0105	LOOP: jmp L1	0001001000100 00000111100010
0106 0107	prn #-5	00001100000000 11111111101100

Decimal Address	Source Code	Binary Machine Code
0108	mov STR[5], STR[2]	00000000101000
0109		00000111110010
0110		00000000010100
0111		00000111110010
0112		00000000001000
0113	sub r1, r4	00000011111100
0114		00000000110000
0115	cmp r3, #sz	00000001110000
0116		00000001100000
0117		00000000001000
0118	bne END	00001010000100
0119		00000111110010
0120	L1: inc K	00000111000100
0121		00001000011110
0122	bne LOOP	00001010000100
0123		00000110100010
0124	END: stop	00001111000000
0125	STR: .string "abcdef"	00000001100001
0126		00000001100010
0127		00000001100011
0128		00000001100100
0129		00000001100101
0130		00000001100110
0131		00000000000000
0132	LIST: .data 6, -9, len	000000000000110
0133		1111111110111
0134		000000000000100
0135	K: .data 22	00000000010110

לאחר סיום העבודה האסמבלר, קבצי הפלט מועברים להמשך עיבוד בשלבי הקישור והטיעינה. לא נדון כאן באופן עבודות שלבי הקישור/טיעינה (כאמור, אלה אינם למימוש בפרויקט זה).

### קבצי קלט ופלט של האסמבלר

בפעולת של האסמבלר, יש להעיר אליו באמצעות ארגומנטים של שורת הפקודה (command line arguments) רשימה של שמות קבצי מקור (אחד או יותר). אלו הם קבצי טקסט, וביהם תכניות בתחביר של שפת האסמבלר, שהוגדרה לעיל. האסמבלר פועל על כל קובץ מקור בנפרד, ויוצר עבורו קובץ מטרה (object) נפרד המכיל את קוד המוכנה. כמו כן האסמבלר יוצר קובץ externals עבור כל קובץ מקור בו יש הצホרות על סמלים חייזוניים, וכן קובץ entries עבור כל קובץ מקור בו יש הצホרות על סמלים כנקודות כניסה.

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

לדוגמה: נניח שתוכנית האסמבלר שלנו נקראת assembler, אז שורת הפקודה הבאה:

assembler x y hello

תפעיל את האסמבלר על הקבצים : .x.as, y.as, hello.as

האסמבלר מגדר שמות לקבצי הפלט המבוססים על שם קובץ הקלט, כפי שהופיע בשורת הפיקודה, בתוספת סיומת שונה. הסיומת ".ob", object, הסיומת ".ent", "עbor קובץ-h-entries", והסיומת ".ext", "עbor קובץ-h-externals".

לדוגמה, בהפעלת האסמבלר באמצעות שורת הפיקודה: assembler x .x.ent .x.ext .x.ob .x.as, וכן קבצי הפלט .x.as, .x.ent, .x.ext, .x.ob, וכן קבצי הפלט entries/externals .x.as, .x.ent, .x.ext, .x.ob, .x.as.

מבנה כל קובץ פלט יתואר בהמשך

## אופן פעולות האסמבילר

נרחיב כאן על אופן פעולות האסמבילר, בנוסף לאלגוריתם השודי שניתנו לעיל.

האסמבילר מחזיק שני מערכים, שיקראו להן מערך הhorאות ומערך הנתונים. מערכים אלו נתונים למשה תמונה של זיכרון המכונה (כל אייר במערך הוא בגודל מילה של המכונה, ככלומר 14 סיביות). במערך הhorאות מכניס האסמבילר את הקידוד של הוראות המכונה שנקרוו במהלך המעבר על קובץ המקור. במערך הנתונים מכניס האסמבילר את קידוד הנתונים שנקרוו מקובץ המקור (שורות הנחיה מסווג 'data' ו-'string').

לאסמבילר יש שני מונחים: מונה הhorאות (IC) ומונה הנתונים (DC). מונחים אלו מצביעים על המקום הבא הפניו במערכות לעיל, בהתאם. כאשרתחליל האסמבילר עבר על קובץ מקור, שני מונחים אלו מאופסים.

בנוסף יש לאסמבילר טבלה, אשר בה נאספות כל התוויות בהן נתקל האסמבילר במהלך המעבר על הקובץ. לטבלה זו קוראים טבלת סמלים (symbol-table). לכל סמל נשמרם שםו, ערכו המספרי, ומאפיינים שונים שצויינו קודם, כגון המיקום (macro) או code או data או relocatable או external).

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

1. שורה ריקה או שורת הערה: האסמבילר מתעלם מהשורה וועבר לשורה הבאה.
2. שורת מאקרו: האסמבילר מכניס את שם המacro לutable הסמלים עם המאפיין macro.
3. שורת הוראה:

האסמבילר מנתח את השורה ומפענח מהי ההוראה, ומהן שיטות המידע של האופרנדים. מספר האופרנדים אותו הוא מחפש נקבע בהתאם להוראה שנמצאה. שיטות המידע נקבעות בהתאם לת לחבר של כל אופרנד, כפי שהסביר לעיל בהגדרת שיטות המידע. למשל, הטו # מצין מידע, תוית מצינית מידע ישיר, שם של אוצר מצין מידע אוגר, ועוד.

אם האסמבילר גילה בשורת ההוראה גם הגדרה של תוית, אז הווית מוכנסת אל טבלת הסמלים. ערך התוית הוא IC+100, והמאפיינים הם code ו-.relocatable

cutת האסמבילר קובע לכל אופרנד את ערכו באופן הבא:

- אם זה אוגר – האופרנד הוא מספר האוגר.
- אם זו תוית (מידע ישיר) – האופרנד הוא ערך התוית כפי שמופיע בטבלת הסמלים (ייתכן והסמל טרם נמצא בטבלת הסמלים).
- אם זה הטו # ואחריו מספר או שם של מאקרו (מידע מיידי) – האופרנד הוא המספר עצמו.
- אם זו שיטות מידע אחרת – ערכו של האופרנד נקבע לפי המפרט של שיטת המידע (ראה תאור שיטות המידע לעיל)

האסמבילר מכניס למערך הhorאות, בכניסה עלייה מצבע מונה הhorאות IC, את קוד המילה הראשונה של ההוראה (בפורמט קידוד כפי שתואר קודם). מילה זו מכילה את קוד הפעלה, ואת מספרי שיטות המידע. ה- IC מוקדם ב-1.

אם זהה ההוראה בעלת אופרנדים (אחד או שניים), האסמבילר "משרין" מקום במערך הhorאות עבור מילוט-המידע הנוספות הנדרשות בהוראה זו, ומקדם את IC בהתאם. כאשר אחד או שני האופרנדים הם בשיטת מידע אוגר-ישיר או מיידי, האסמבילר מקובד גם את המילים הנוספות הרלוונטיות במערך הhorאות.

נזכר שם יש רק אופרנד אחד, ככלומר אין אופרנד מקור, הסיבות של שיטת המיעון של אופרנד המקור יכולו תמיד 0, מכיוון שאין רלוונטיות. בדומה, אם זהה הוראה ללא אופרנדים, (rts, stop, אזי הסיבות של שיטת המיעון של שני האופרנדים יכולו 0).

#### 4. שורת הנחיה:

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

##### I. '.data'

האסמבלר קורא את רשימת המספרים, המופיע לאחר '.data', מכניס כל מספר אל מערך הנתונים, ומقدم את מצביע הנתונים DC באחד עבור כל מספר שהוכנס. נשים לב שגם שם של מאקרו יכול לשמש במקום מספר.

אם בשורה '.data' מוגדרת גם תווית זו מוכנסת לטבלת הסמלים. ערך התווית הוא ערך מונה הנתונים DC שלפניהם הכנסת המספרים למערך. המאפיינים של התווית הם data ו-.relocatable

##### II. '.string'

הטיפול ב-'string' דומה ל-'data', אלא שקודם ascii של התווים הם אלו המוכנסים אל מערך הנתונים (כלתו בכינסה נפרדת). לאחר מכן מוכנס הערך 0 (המצוין סוף מהorzot) אל מערך הנתונים. מונה הנתונים מקודם באורך המחרוזת + 1 (גם האפס בסוף המחרוזת יופס מקום).

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

##### III. '.entry'

זהי בקשה לאסמבלר להכניס את התווית המופיע כאופרנד של '.entry' אל קובץ ה-entries. האסמבלר רושם את הבקשה ובסיום העבודה, התווית הניל תירשם בקובץ ה-entries.

##### IV. '.extern'

זהי הקריאה על סמל (תווית) המוגדר בקובץ אחר, ואשר קטע האסמבלי בקובץ הנוכחי עושים בו שימוש. האסמבלר מכניס את הסמל אל טבלת הסמלים, עם הערך 0 (הערך האמתי לא ידוע, וייקבע רק בשלב הקישור), והמאפיין external. לא ידוע באיזה קובץ נמצאת הגדרת הסמל (וגם אין זה משנה עבור האסמבלר).

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

בסוף המעבר הראשון, האסמבלר מעדכן בטבלת הסמלים כל סמל המופיע כ-'data', על ידי הוספת IC+100 (עשורוני) לערכו של הסמל. הסיבה לכך היא שבתמונה הכלולת של קוד המcona, הנתונים מופרדים מההוראות, וכל הנתונים נדרשים להופיע אחרי כל ההוראות. סמל מסווג data הוא למעשה תווית באוצר הנתונים, והעדכו מוסיף לערך הסמל (כלומר כתובתו בזיכרו) את האורך הכללי של קידוד כל ההוראות, בתוספת כתובות התחלת הטעינה של הקוד, שהוא 100.

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

במעבר השני, האסמבלר מקודד באמצעות טבלת הסמלים את כל המילים במערך ההוראות שטרם קודדו במעבר הראשון. אלו הן מילים שצרכו להכיל כתובות של תוויות (שדה ה-A,R,E,Bmilim אלה יהיה 10 או 01).

## פורמט קובץ ה-object

כזכור, האסמבלר בונה את תומנת הזיכרון של התכנית כך שקידוד ההוראה הראשונה מקובץ האסמבלי ייכנס למן 100 (בסיס עשרוני) בזיכרון, קידוד ההוראה השנייה ייכנס למן העוקב אחרי ההוראה הראשונה (תלו依 במספר המילים של ההוראה הראשונה), וכך הלאה עד להוראה الأخيرة.

אחרי ההוראה الأخيرة, מכניםים לתומנת הזיכרון את קידוד איזור הנתונים, שנבנה על ידי `ההנחיות' .data.string`. זהה הסיבה בגללה יש לעמוד בטבלת הסמלים, בסוף המעבר הראשון, את ערכי הסמלים עם המאפיין `.data`.

נשים לב שבגמר המעבר השני, אופרנד של הוראה המתיחס לסמל שהוגדר באותו קובץ, כבר מקודד כך שייצביע על המקום המתאים בתומנת הזיכרון שבונה האסמבller.

cut האסמבller יכול להעביר את תומנת הזיכרון בשלמותה לתוכן קובץ פלט (קובץ ה-object).

השורה הראשונה בקובץ ה-object היא "cotract", המכילה שני מספרים (בסיס עשרוני) : הראשון הוא האורך הכלול של קטע ההוראות (ב밀ות זיכרון), והשני הוא האורך הכלול של קטע הנתונים (ב밀ות זיכרון). בין שני המספרים יש רווח אחד.

השורות הבאות בקובץ מכילות את תומנת הזיכרון. בכל שורה שני ערכים : כתובות של מילה בזיכרון, ותוכן המילה. הכתובת תירשם בסיסי עשרוני באربע ספרות (כולל אפסים מוביילים). תוכן המילה יירשם בסיס 4 "מיוחד" (ראה להלן) בשבע ספרות (כולל אפסים מוביילים). בין שני הערכים בכל שורה יפריד רווח אחד.

בסיס 4 רגיל	0	1	2	3
בסיס 4 מיוחד	*	#	%	!

קובץ object לדוגמה, כפי שאמור להיבנות על ידי האסמבller, נמצא בהמשך.

## פורמט קובץ ה-entries

קובץ ה-entries בנייתו טקסט. כל שורה מכילה שם של סמל שהוגדר כ- entry ואת ערכו, כפי שנמצא בטבלת הסמלים. הערכים מיוצגים בסיס עשרוני.

## פורמט קובץ ה-externals

קובץ ה-externals בנייתו אף הוא משוראות טקסט. כל שורה מכילה שם של סמל שהוגדר external וכתובות בקוד המכונה בה יש קידוד של אופרנד המתיחס לסמל זה. כMOVן שייתכן ויש מספר כתובות בקוד המכונה בהם מתיחסים לאותו סמל חיצוני. כל התייחסות כזו תהיה שורה נפרדת בקובץ ה-externals. הכתובות מיוצגות בסיס עשרוני.

נדגים את קבצי הפלט שמייצר האסמבller עבור קובץ מוקור בשם `sk.nntwo להלן`.

```
; file ps.as
```

```
.entry LIST
.extern W
.define sz = 2
MAIN:    mov   r3, LIST[sz]
LOOP:    jmp   W
        prn   #-5
        mov   STR[5], STR[2]
        sub   r1, r4
```

```

        cmp  K, #sz
        bne W
L1:    inc  L3
.entry LOOP
        bne  LOOP
END:   stop
.define len = 4
STR:    .string "abcdef"
LIST:   .data 6, -9, len
K:      .data 22
.extern L3

```

להלן הקידוד הבינארי המלא (תמונה הזיכרון) של קודץ המקור, כפי שנבנה במעבר הראשון והשני.

Decimal Address	Source Code	Binary Machine Code
0100	MAIN: mov r3, LIST[sz]	00000000111000
0101		00000001100000
0102		00001000010010
0103		000000000001000
0104	LOOP: jmp W	00001001000100
0105		0000000000000001
0106	prn #-5	00001100000000
0107		1111111101100
0108	mov STR[5], STR[2]	00000000101000
0109		00000111110110
0110		00000000010100
0111		00000111110110
0112		000000000001000
0113	sub r1, r4	00000011111100
0114		000000000110000
0115	cmp K, #sz	00000001010000
0116		00001000011110
0117		000000000001000
0118	bne W	00001010000100
0119		0000000000000001
0120	L1: inc L3	00000111000100
0121		0000000000000001
0122	bne LOOP	00001010000100
0123		00000110100010
0124	END: stop	00001111000000
0125	STR: .string "abcdef"	00000001100001
0126		00000001100010
0127		00000001100011
0128		00000001100100
0129		00000001100101
0130		00000001100110
0131		0000000000000000
0132	LIST: .data 6, -9, len	000000000000110 11111111110111
0133		000000000000100
0134		
0135	K: .data 22	00000000010110

להלן תוכן קבצי הפלט של הדוגמה.

:הקובץ ps.ob

```

25 11
0100  ****!%
0101  * * # %
0102  * * % * # * %
0103  * * * * %
0104  * * % # * #
0105  * * * * * #
0106  * * ! * * *
0107  ! ! ! % ! *
0108  * * * * % ! *
0109  * * # ! ! # %
0110  * * * * # #
0111  * * # ! ! # %
0112  * * * * %
0113  * * * ! ! !
0114  * * * * ! *
0115  * * * # #
0116  * * % * # ! %
0117  * * * * #
0118  * * % % * #
0119  * * * * * #
0120  * * # ! * #
0121  * * * * * #
0122  * * % % * #
0123  * * # % % * %
0124  * * ! ! *
0125  * * * # % * #
0126  * * * # % * %
0127  * * * # % * !
0128  * * * # % #
0129  * * * # % #
0130  * * * # % #
0131  * * * * *
0132  * * * * %
0133  ! ! ! ! !
0134  * * * * #
0135  * * * # %

```

:הקובץ ps.ent

```

LOOP 0104
LIST 0132

```

כל המספרים בבסיס עשרוני

:הקובץ ps.ext

```

W      0105
W      0119
L3     0121

```

כל המספרים בbasis עשרוני

לתשומתך : 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 מסוג ent. או ext. כל שורה עומדת בפני עצמה.

## סיכום והנחיות כלליות

- גודל תוכנית המקור הניתנת כקלט לאסטבלר אינו ידוע מראש, ולכן גם גודלו של קוד המוכנה אינו צפוי מראש. אולם בכך להקל במיומוש האסטבלר, מוטר להניח גודל מסוימלי. לפיכך יש אפשרות להשתמש במערכות לאחסן תמונות קוד המוכנה בלבד. כל מבנה נתונים אחר (למשל טבלת הסמלים), יש למשול באופן ייעיל וחסכוני (למשל באמצעות רשימה מקוורת והקצתה זיכרון דינامي).
- השמות של קבצי הפלט צריכים להיות תואמים לשם קובץ הקלט, למעט הסיומות. למשל, אם קובץ הקלט הוא asprog, ob, prog.ext, prog.ent.
- מתכונת הפעלת האסטבלר צריכה להיות כפי הנדרש בממ"ן, ללא שינויים כלשהם. ככלומר, ממשך המשתמש יהיה אך ורק באמצעות שורת הפוקודה. בפרט, שמות קבצי המקור יועברו לתכנית האסטבלר כארגומנטים בשורת הפוקודה. אין להוסיף תפריטי קלט אינטראקטיביים, חלונות גרפיים למיניהם, ועוד'.
- יש להקפיד לחלק את מיומוש האסטבלר למספר מודולים (קבצים בשפט C) לפי משימות. אין לרכז משימות מסוימים 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 (קוד הפעולה, שיטות המיעון החוקיקות לכל פעולה, ועוד').
- יש להקפיד ולתעד את המיומוש באופן מלא וברור, באמצעות הערות מפורחות בקוד.
- יש לאפשר תווים לבנים עודפים בקובץ הקלט בשפט אסטבלרי. למשל, אם בשורת הוראה יש שני אופרדים המופרדים בפסיק, אזי לפני ואחריו הפסיק מותר להיות רווחים וטאבבים בכל 경우에는. גם לפני ואחריו שם הפעולה. מותרות גם שורות ריקות. האסטבלר יתעלם מתווים לבנים מיותרים (כלומר ידלג עליהם).
- הקלט (קוד האסטבלר) עלול להכיל שגיאות תחביריות. על האסטבלר לגלות ולדווח על כל השגיאות בקלט. אין לעצור את הטיפול בקובץ קלט לאחר גילוי השגיאה הראשונה. יש להדפיס לפחות הודעות מפורטות ככל הנינתן, כדי שאפשר יהיה להבין מה והיכן כל שגיאה. כמובן שם קובץ קלט מכיל שגיאות, אין טעם להפיק עבورو את קבצי הפלט (ob, ext, ent).

**גם ונשלם פרק ההסברים והגדרת הפרויקט.**

**בשאלות ניתן לפנות לקבוצת הדיוון באתר הקורס, ועל כל אחד מהמנחים בשעות הקבלה שלהם.**

להזיכרכם, באפשרותו של כל סטודנט לנחות לכל מנהה, לאו דווקא למנהל הקבוצה שלו, לקבלת עזרה. שוב מומלץ לכל אלה שטרם בדקו את התכנים באתר הקורס לעשות זאת. נשאלות באתר זה הרבה שאלות בנושא חומר הלימוד והממ"נים, והתשובות יכולות להופיע בכלום.

לתשומתיכם: לא תננו דחיה בהגשת הממ"ן, פרט למקרים מיוחדים כגון מילואים או מחלת. במקרים אלו יש לבקש ולקבל אישור מראש מצוות הקורס.

**ב ה צ ל ח ה !**