

# מטלת מנהה (ממ'ו) 14

הקורס: 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטרת: פרויקט גמר

משקל המטרת: 31 נקודות (חוובה)

מספר השאלות: 1

מועד אחרון להגשת: 29.3.2020

סמסטר: 2020א'

## קיימות שתי חלופות להגשת מטלות:

• שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס

• שליחת מטלות באמצעות דואר אלקטרוני - אישור המנהה בלבד

## הסבר מפורט ב" ניהול הגשת מטלות מנהה"

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

עליכם כתוב תוכנת אסמבילר, עבור שפת אסמבלי שתוגדר בהמשך. הפרויקט ייכתב בשפת C

עליכם להגיש:

1. קבצי המקרה של התוכנית שכתבתם (קבצים בעלי הסיומת .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 החודעות שモציא הקומפיילר, כך שהתוכנית תתקמפל ללא כל העורות או זהירות.

4. דוגמאות הרצה (קלט ופלט) :

א. קבצי קלט בשפת אסמבלי, וקבצי הפלט שנוצרו מהפעלת האסמבילר על קבצי קלט אלה. יש להציג שימוש במגוון הפעולות וטיפוסי הנתונים של שפת האסמבילר.

ב. קבצי קלט בשפת אסמבילר המדינים מגוון רחב של סוגים של אסמבלי (ולכן לא נוצרים קבצי פלט), ותდפסי המשך המראים את הودעות השגיאה שמוציאה האסמבילר.

בשל גודל הפרויקט, עליכם לחלק את התוכנית למספר קבצי מקור, לפי מישימות. יש להקפיד שקוד המקור של התוכנית יעמוד בקריטריונים של בהירות, קריאות וכתיבה נאה ונוונית.

זכור מספר היבטים חשובים של כתיבת קוד טוב:

1. הפשטה של מבני הנתונים: רצוי (כלל האפשר) להפריד בין הגישה למבנה הנתונים לבין הIMPLEMENTATION של מבני הנתונים. כך, למשל, בעת כתיבת פונקציות לטיפול בטבלה, אין זה מעניינן של משתמשים בפונקציות אלה, האם הטבלה ממומשת באמצעות מערך או באמצעות רשיימה מקוורת.

2. קריאות הקוד: יש להשתמש במסות שימושיים למשתנים ופונקציות. כמו כן, רצוי להגדיר קבועים רלוונטיים תוך שימוש בהנחתה #define, ולהימנע מ"מספרי קסם", שימושיהם נהירה לכם בלבד. יש לעורוך את הקוד באופן מסודר: הזחות עיקריות, שורות ריקות להפרדה בין קטעי קוד, וכו'.

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

**הערה:** תוכנית "עובדת", דהיינו תוכנית שمبرכעת את הדרוש ממנה, אינה לכשעצמה עovable לציוון גובה. כדי לקבל ציוון גבוה, על התוכנית לעמוד בקריטריונים של כתיבה ותיעוד ברמה טובה, כמתואר לעיל, אשר משקלם המשותף מגיעה עד לכ- 40% משקל הפרויקט.

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

מומלץ לעבוד בזוגות. אין לעובד בצוותים גדולים יותר. **פרויקט שיוגש על ידי שלשה או יותר, לא יבחן ולא יקבל ציוון.** חובה שסטודנטים, הבוחרים להגיש יחד את הפרויקט, יהיו **שייבים** לאותה קבוצת הנחיה. הציוון יהיה זהה לשני הסטודנטים.

מומלץ לקרוא את הגדרת הפרויקט עם ראשונה ברצף, לקבלת תמורה כללית לגבי הנדרש, ורק לאחר מכן לקרוא שוב בקורס 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 ביארוי. קוד זה מאוחסן בזיכרון, וראה כמו רצף של ספרות ביארויות. יחידת העיבוד המרכזית - היע"מ (CPU) - יודעת לפרק את הרצף הזה לקטעים קטנים בעלי משמעות: הוראות, מענים ונתונים.

למעשה, זיכרונו המחשב כולל הוא אוסף של סיביות, שנוהגים לראותו כמרקיבות יחידות בעלות אורך קבוע (BITS, MILLS). לא ניתן להבחין, בין שайינה מיומנת, בהבדל פיסי כלשהו בין אותן חלק בזיכרון שבו נמצאת תוכנית לבין שאור הזיכרון.

**יחידת העיבוד המרכזית (היע"מ)** יכולה לבצע מגוון פעולות פשוטות, הנקראות הוראות המכונה, ולשם לכך היא משתמש באוגרים (registers) הקיימים בתוך היע"מ, ובזיכרון המחשב.  
**donegmaot:** העברת מספר מתא בזיכרון לאור בע"מ או בחזרה, הוספה 1 למספר הנמצא באור, בדיקה האם מס' המתאים באור שווה לאפס, חיבור וחיסור בין שני אוגרים, וכו'. הוראות המכונה ושילובים שלهنן הן המרכיבות תוכנית כפי שהיא טעונה לו זיכרון בזמן ריצתה. כל תוכנית מקור (התוכנית כפי שנכתבה בידי המפתח), תורגמת בסופו של דבר באמצעות תוכנה מיוחדת לזרעה סופית זו.

היע"ם יודע לבצע קוד שנמצא בפורמט של **שפת מכונה**. זהו רצף של ביטים, המהווים קידוד ביןארוי של סדרת הוראות המכונה המרכיבות את התוכנית. קוד כזה אינו קריא למשתמש, ולכן לא נוח לקוד (או לקרוא) תוכניות ישירות בשפת מכונה. **שפת אסמבלי** (assembly language) היא שפת תכנות מאפרשת לייצג את הוראות המכונה בזרחה 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 שחקרא אסמבלי (assembler).

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

כל מודול של יע"מ (כלומר לכל אירוגון של מחשב) יש שפת מכונה יעודית משלו, ובהתאם גם שפת אסמבלי יעודית משלו. לפיכך, גם האסמבלי (כל התרגומים) הוא יודע ושותה לכל יע"מ.

תפקידו של האסמבלי הוא לבנות קוד המכיל קוד מכונה, מקובץ נתון של תוכנית הכתובת בשפת אסמבלי. זהו השלב הראשון במסלול אותו עברות התוכנית, עד לקבלת קוד המוכן לריצה על חומרה המחשב. השלבים הבאים הם קישור (linkage) וטעינה (loading), אך בהם לא נעסק במMING' זה.

המשמעותה בפרויקט זה היא לכתוב אסמבלי (כלומר תוכנית המתרגם לשפת מכונה), עבור שפת אסמבלי שנגידר כאן במיוחד לצורך הפרויקט.

**لتשומת לב:** בהסבירים הכלליים על אופן העבודה תוכנת האסמבלי, תהיה מדי פעם התייחסות גם לעובdot שלבי הקישור והטעינה. התיחסויות אלה נועדו על מנת לאפשר לכם להבין את המשך תהליך העיבוד של הפלט של תוכנת האסמבלי. אין לטעות: עלייכם לכתוב את תוכנית האסמבלי בלבד. אין לכתוב את תוכניות הקישור והטעינה!!!

המחשב הדמיוני וסתת האסמבלי

הערה: תואור מודל המחשך להלן הוא חלקי בלבד. ככל שהוחוץ לביצוע המשימות בפרויקט.

**”חומרה”:**

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

למיעבד 8 אוגרים כלליים, בשמות: r<sub>7</sub>, r<sub>4</sub>, r<sub>5</sub>, r<sub>6</sub>, r<sub>1</sub>, r<sub>2</sub>, r<sub>3</sub>, r<sub>0</sub>. הסבירתי הכי פורחות משמעויות תצווין כסיבית מס' 0, והסבירתי גודל של כל אוגר הוא 15 טבניות. שמות האוגרים כתובים תמיד עם אות 'z', קטןנה.

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

גודל הזיכרון הוא 4096 תאים, בכתובות 0-4095 (בבסיס עשרוני), וכל תא הוא בגודל של 15 סיביות.unta  
لتא בזיכרון נקרא גם בשם "מילה". הшибיות בכל מילה ממופרות כמו באוגר.

האריתמטיקה נעשית בשיטת המשלים ל-2 (2's complement). כמו כן יש תמייה בתווים (characters), המוצגים בקוד ascii.

#### מבנה הוראת מכונה:

כל הוראות מכונת מקודדת במספר מילויות זיכרונות רצופות, החל מ밀ילה אחת ועד למקסימום שלוש מיללים, הכל בהתאם לשיטות המיעוון בהן נעשה שימוש (ראו בהמשך).

בקובץ הפלט המכיל את קוד המכונה שבונה האסמבולר, כל מילה תקודד בסיס אוקטלי (ראו פרטיס לגביו בדף פלט בהמשך).

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

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				שיטות מייעון אופרנד מקור					שיטות מייעון אופרנד יעד					השדה A,R,E
opcode	ש	ש	ש	ש	ש	ש	ש	ש	ש	ש	ש	A	R	E
	,	,	,	,	,	,	,	,	,	,	,			
	ט	ט	ט	ט	ט	ט	ט	ט	ט	ט	ט			
	ה	ה	ה	ה	ה	ה	ה	ה	ה	ה	ה			
	3	2	1	0	3	2	1	0						

**סיביות 11-14:** בambil הראונה של ההוראה סיביות אלה מהוות את קוד-הפעולה (opcode). במודל המכונה שלנו יש 16 קודי פעולה. כל קוד-פעולה (opcode) מיוצג בשפת אסמבלי באופן סימבולי על ידי שם-פעולה.

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

שם הפעולה נכתב תמיד באותיות קטנות בלבד. פרטים על הפעולות השונות יובאו בהמשך.

**סיביות 3-6:** מקודדות את שיטת המיעון של אופרנד היעד (destination operand). לכל שיטת מיעון יש סיבית נפרדת, שערכה 1 אם אופרנד היעד נתון בשיטה זו, ואחרת ערך הסיבית 0. אם אין בהוראה אופרנד יעד, כל ארבע הסיביות מאופסות.

**סיביות 7-10:** מקודדות את שיטת המיעון של אופרנד המקור (source operand). לכל שיטת מיעון יש סיבית נפרדת, שערכה 1 אם אופרנד המקור נתון בשיטה זו, ואחרת ערך הסיבית 0. אם אין בהוראה אופרנד מקור, כל ארבע הסיביות מאופסות.

**סיביות 0-2 (השדה 'E,A,R,E':)**: אפיקן של תפקיד השדה 'A,R,E' בקוד המכונה יובא בהמשך. במילה הראשונה של כל הוראה, ערך הסיבית A תמיד 1, ושתי הסיביות האחרות מאופסות.

لتשומת לב: **השדה 'E,A,R,E'** מנוסך לכל אחת מהמילים בקידוד ההוראה (ראו פירוט של שיטות המיעון בהמשך).

#### שיטות מיעון:

בשפה האסטטבלי שלנו קיימות ארבע שיטות מיעון, המסומנות במספרים 0,1,2,3. השימוש בשיטות מיעון מצריך קידוד של מילוט-מידע נוספת בקוד המכונה של כל הוראות מכונה. כל אופרנד של ההוראה דורש מילה אחת נוספת.

כאשר בהוראה יש שני אופרנדים, קודם ו紧跟ם מילת-המידע הנוספת של האופרנד הראשון (אופרנד המקו), ולאחריה מילת-המידע הנוספת של האופרנד השני (אופרנד היעד). קיימים גם מקרים מיוחדים בו קידוד שני האופרנדים נעשה באמצעות מילת-מידע אחת משותפת לשני האופרנדים.

כל מילת-מידע נוספת של ההוראה מקודדת באחד משלשה סוגים של של קידוד. סיביות 0-2 של כל מילת-מידע הן השדה 'A,R,E', המציין מהו סוג הקידוד של המילה. לכל סוג קידוד יש סיבית נפרדת, שערכה 1 אם מילת-המידע נתונה בסוג קידוד זה, ואחרת ערך הסיבית הוא 0.

- סיבית 2 (הסיבית A) מציינת שקיידוד המילה הוא מוחלט (Absolute), ואינו מצריך שינוי בשלב ה קישור והטעינה.
- סיבית 1 (הסיבית R) מציינת שהקיידוד הוא של כתובות פנימיות הנינטת להזזה (Relocatable), ומצריך שינוי בשלב ה קישור והטעינה.
- סיבית 0 (הסיבית E) מציינת שהקיידוד הוא של כתובות חיצונית (External) , ומצריך שינוי בשלב ה קישור והטעינה.

הסביר על התפקיד של השדה 'E' בקוד המכונה יבוא בהמשך.

ערך השדה 'A,R,E' הנדרש בכל שיטות מייען מופיע בתיאור שיטות המייען להלן.

מספר	שיטת המייען	תוכן מילת-המידע הנוספת	אופן כתיבת האופרנד	דוגמה
0	מייען מיידי	밀ת-מידע נוספת של ההוראה מכילה את האופרנד עצמו, שהוא מספר שלם בסיס עשרוני. הסיביות 0-2 של מילת המידע הן השדה A,R,E. במייען מיידי, ערך הסיבית A הוא 1, ושתי הסיביות האחרות מאופסות.	האופרנד מתחיל בトー # ולאחריו ובצמוד אליו מופיע מספר שלם בסיס עשרוני.	mov #-1,r2  בדוגמה זו האופרנד הראשון של הפקודה (אופרנד המקור) נתון (ארך -1- אל אוגר 2 ז')
1	מייען ישיר	밀ת-מידע נוספת של ההוראה מכילה כתובות זיכרון. המילה כתובות זו בזיכרון היא האופרנד. הכתובת מיוצגת כמספר ללא סימן ברוחב של 12 סיביות, בסיביות 3-14 של מילת המידע.  הסיביות 0-2 במילת המידע הן השדה A,R,E. במייען ישיר, ערך הסיביות האלה תלוי בסוג הכתובת השמוכה בסיביות 14-3. אם זהה כתובות שמייצגת שורה בקובץ המקור הנוכי (כתובות פנימיות), ערך הסיבית R הוא 1, ושתי הסיביות האחרות מאופסות. ואילו אם זהה כתובות שמייצגת שורה בקובץ מקור אחר של ה恬נית (כתובות חיצונית), ערך הסיבית E הוא 1, ושתי הסיביות האחרות מאופסות.	האופרנד הוא <u>תוויות</u> שכבר הוגדרה, או שתוגדר בהמשך הקובץ. ההגדרה נעשית על ידי כתיבת תווית בתחילת הנקה 'data.string', או 'string.' או בתחילת הוראה של הנקיות, או באמצעות אופרנד של הנקה 'extern.'	x: .data 23  ההוראה:  dec x  מקטינה ב-1 את תוכן המילה שבכתבות x בזיכרון (ה" משתנה " x).

מספר	שיטת המיעון	תוכן מילת-המידע הנוספת	אופן כתיבת האופרנד	דוגמה
2	מייעון אוגר עקיף	<p>שיטת מייעון זו משמשת לגישה לזכרון באמצעות מצביע שנמצא באוגר. תוכן האוגר הוא כתובות בזיכרון, והmillion הינה כתובות בזיכרון היא האופרנד. הכתובות מיצגתו באוגר כמספר לא סימן ברוחב של 15 סיביות.</p> <p>אם האופרנד הוא אופרנד יעד, מילת-מידע נוספת נוספת של הפוקודה תכיל בסיביות 5-3 את מספרו של האוגר שימושו כמצבי. ואילו אם האוגר הוא אופרנד מקור, מספר אוגר יקודד בסיביות 6-8 של מילת-המידע הנוספת.</p> <p>הסיביות 0-2 של מילת המידע הן השדה A,R,E. במייעון אוגר עקיף, ערך הסיבית A הוא 1, ושתי הסיביות האחרות מאופסות.</p> <p>אם בפוקודה יש שני אופרנדים, וכל אחד מהם בשיטת מייעון אוגר עקיף או בשיטת מייעון אוגר ישיר (ראו בהמשך), שני האוגרים יחלקו מילת-מידע אחת משותפת, כאשר סיביות 3-5 ייכלו את מספר אוגר היעד, וסיביות 6-8 את מספר אוגר המקור. השדה A,R,E יהיה כמפורט לעיל.</p> <p>סיביות במילת-המידע שאינן בשימוש ייכלו 0.</p>	<p>האופרנד מתחל בטו *</p> <p>ולאחריו ובצמוד אליו מופיע שם של אוגר.</p>	<p>inc *r1 בדוגמה זו, ההוראה inc מגדילה ב-1 את תוכן המילה בזיכרון עליה מצבי האוגר 1, כלומר המילה שכותבתה נמצאת באוגר 1.</p> <p>דוגמה נוספת : mov *r1,*r2 בדוגמה זו, ההוראה mov מעתקה את תוכן המילה בזיכרון עליה מצבי האוגר 1 ז' אל המילה בזיכרון עליה מצבי האוגר 2. שני האופרנדים בדוגמה זו הם בשיטת מייעון אוגר עקיף, וכן שני האוגרים יקודדו במילת-מידע נוספת אחת משותפת.</p>
3	מייעון אוגר ישיר	<p>האופרנד הוא שם של אוגר. אופרנד כאופרנד יעד, מילת-מידע נוספת של הפוקודה תכיל בסיביות 5-3 את מספרו של האוגר. ואילו אם האוגר משמש כאופרנד מקור, מספר האוגר יקודד בסיביות 6-8 של מילת-המידע.</p> <p>הסיביות 0-2 של מילת המידע הן השדה A,R,E. במייעון אוגר ישיר, ערך הסיבית A הוא 1, ושתי הסיביות האחרות מאופסות.</p> <p>אם בפוקודה יש שני אופרנדים, וכל אחד מהם בשיטת מייעון אוגר ישיר או בשיטת מייעון אוגר עקיף (ראו לעיל), שני האוגרים יחלקו מילת-מידע אחת משותפת, כאשר סיביות 3-5 ייכלו את מספר אוגר היעד, וסיביות 6-8 את מספר אוגר המקור. השדה A,R,E יהיה כמפורט לעיל.</p> <p>סיביות במילת-המידע שאינן בשימוש ייכלו 0.</p>	<p>האופרנד הוא שם של אוגר.</p>	<p>clr r1 בדוגמה זו, ההוראה clr מפסת את תוכן האוגר ז'. דוגמה נוספת : mov r1,*r2 בדוגמה זו, ההוראה mov מעתקה את תוכן האוגר 1 ז' אל המילה בזיכרון 2. אליה מצבי האוגר 2. אופרנד המקור 1 ז' נתון בשיטת מייעון אוגר ישיר, ואופרנד היעד 2 ז' בשיטת מייעון אוגר עקיף, וכן שני האוגרים יקודדו במילת-מידע נוספת אחת משותפת.</p>

## מפורט הוראות המכונה:

ההוראות המכונה מתחולקות לשלוש קבוצות, לפי מספר האופרנדים הדרושים לפעולה.

**קבוצת ההוראות הראשונה:**  
אלן הוראות הדורשות שני אופרנדים.

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

הסבר הדוגמה	דוגמה	הפעולה המתבצעת	opcode	הוראה
העתק את תוכן המשתנה A (המילה שבכותרת A בזיכרון) אל אוגר r1.	mov A, r1	מבצע העתקה של האופרנד (source), אופרנד המקור (הראשון, אופרנד השני, אופרנד היעד (destination) (בהתאם לשיטת המיעון).	0	mov
אם תוכן המשתנה A זהה לתוכנו של אוגר r2 אז דגל האפס, Z, באוגר הסטטוס PSW (יודלק, אחרת הדגל יאפס.	cmp A, r1	מבצעת "השוואה" בין שני האופרנדים שלה. אופן ההשוואה: תוכן אופרנד המקור (הראשון) מופחת מהתוך אופרנד המקרה (השני), ללא שמירה תוצאה החישוב. פועלות החישוב מעದכנת דגל בשם Z ("דגל האפס") באוגר הסטטוס (PSW).	1	cmp
אוגר r0 מקבל את תוצאה החיבור של תוכן המשתנה A ותוכנו הנוכחי של r0.	add A, r0	אופרנד היעד (השני) מקבל את תוצאה החיבור של אופרנד המקור (הראשון) והיעד (השני).	2	add
אוגר r1 מקבל את תוצאה החישוב של הערך 3 מתוכנו הנוכחי של האוגר r2.	sub #3, r1	אופרנד היעד (השני) מקבל את תוצאה החישוב של אופרנד המקור (הראשון) מופרנד היעד (השני).	3	sub
המען שמייצגת התוויות HELLO מוצב לאוגר r1.	lea HELLO, r1	lea הוא קיצור (ראשי תיבות) של load effective address. פעולה זו מצייבה את המען בזיכרון המצויג על ידי התוויות שבאופרנד הראשון (המקורה, אל אופרנד היעד (השני)).	4	lea

## קבוצת ההוראות השנייה:

אלן הוראות הדורשות אופרנד אחד בלבד. אופן הקידוד של האופרנד הוא כמו של אופרנד היעד בפוקודה עם שני אופרנדים. במקרה זה, השדה של אופרנד המקור (סיביות 10-7) במילה הראשונה בקידוד ההוראה הינו חסר משמעות, ולפיכך יכיל 00.

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

הסבר הדוגמה	דוגמה	הפעולה המתבצעת	opcode	הוראה
r2 ← 0	clr r2	אפסת תוכן האופרנד	5	clr
r2 ← not r2	not r2	היפוך ערכי הסיביות באופרנד (כל סיבית שערכה 0 תהפוך ל-1 ולחיפך 1 ל-0).	6	not
r2 ← r2 + 1	inc r2	הגדלת תוכן האופרנד באחד.	7	inc
C ← C - 1	dec C	הקטנת תוכן האופרנד באחד.	8	dec

הסבר הדוגמה	דוגמאות	הפעולה המתבצעת	opcode	הוראה
PC $\leftarrow$ LINE מצביע התוכנית מקבל את המعنן המוצג על ידי הכתובית LINE, ויפריך הפוקודה הבאה שתתבצע תהיה במען זה.	jmp LINE	קפיצה (הסתעפות) בלתי מותנית אל הוראה שנמצאת במען המוצג על ידי האופרנד. לעומת זאת מביצוע הוראה, מביע תוכנית (PC) קיבל את ערך אופרנד היעד.	9	jmp
אם ערך הדגל Z באוגר הסטטוס (PSW) הינו 0, אז: PC $\leftarrow$ LINE	bne LINE	bne הוא קיצור (ראשי תיבות) של .branch if not equal (to zero). זה הוראת הסתעפות מותנית. מביע התוכנית (PC) קיבל את ערך אופרנד היעד אם ערכו של הדגל Z באוגר הסטטוס (PSW) הינו 0. כזכור, הדגל Z נקבע בפקודת .cmp.	10	bne
קוד האסצ'י של התו הנקרא מהקלט ייכנס לאוגרא.z.	red r1	קריאה של تو מהקלט הסטנדרטי (stdin) אל האופרנד.	11	red
התו אשר קוד האסצ'י שלו נמצא באוגרא z יודפס לפלט הסטנדרטי.	prn r1	הדפסת התו הנמצא באופרנד, אל הפלט הסטנדרטי (stdout).	12	prn
push(PC) PC $\leftarrow$ FUNC	jsr FUNC	קריאה לשגרה (סברוטינה), מביע התוכנית (PC) הוכחי נדחף לתוך המחסנית שזובירון המחשב, והאופרנד מוכנס ל-PC.	13	jsr

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

ההוראות השויות לקבוצה זו הן : .rts, stop

הסבר הדוגמה	דוגמאות	הפעולה המתבצעת	opcode	הוראה
PC $\leftarrow$ pop()	rts	חזרה משיגרת. הערך שנמצא בראש המחסנית של המחשב מצוי מן המחסנית, ומוכנס אל מביע התוכנית (PC).	14	rts
התוכנית עצרת	stop	עצירת ריצת התוכנית.	15	stop

#### מבנה שפת האסמבלי :

שפת האסמבלי בנויה ממשפטים (statements). קובץ בשפת אסמבלי מורכב משורות המכילות משפטי של השפה, כאשר כל משפט מופיע בשורה נפרדת. לעומת זאת המפריד בין משפט לבין המשפט הבא הינו התו 'ת' (שורה חדשה).

אורכה של שורה בקובץ המקור הוא 80 תווים לכל היתר (לא כולל התו ט).

יש ארבעה סוגי משפטיים (שורות) בשפת האסמבלי, והם :

סוג המשפט	הסבר כללי
משפט ריק	זהי שורה המכילה אך ורף תווים לבנים (whitespace), ככלומר רק את התווים ' ' ו- '\n' (רווחים וטאבים). ייתכן ובשורה אין אף تو (למעט התו מ), כלומר השורה ריקה.
משפט הערה	זהי שורה בה התו הראשון הינו ';' (נקודה פסיק). על האסמבול להתעלם לחלוטין משורה זו.
משפט הנחיה	זהו משפט המנחה את האסמבול מה עליו לעשות כשהוא פועל על תכנית המקור. יש מספר סוגי של משפטי הנחיה. משפט הנחיה עשוי לגורם להקצאת זיכרון ואתחלול משתנים של התכנית, אך הוא אינו מייצר קידוד של הוראות מכונה המיעודות לביצוע בעת ריצת התכנית.
משפט הוראה	זהו משפט המייצר קידוד של הוראות מכונה לביצוע בעת ריצת התכנית. המשפט מורכב ממש של הוראה שעל המעבד לבצע, ותיאור האופרגדים של הוראה.

cut נפרט יותר לגבי סוגי המשפטים השונים.

#### משפט הנחיה:

משפט הנחיה הוא בעל המבנה הבא :

בתחילת המשפט יכול להופיע הגדרה של תווית (label). לתווית יש לחבר חוקי שיתואר בהמשך. התווית היא אופציונאלית.

לאחר מכן מופיע שם הנחיה. לאחר שם הנחיה יופיע פרמטרים (מספר הפרמטרים בהתאם להנחיה). לאחר מכן מופיע שם של הנחיה מתייחס באתיות קטנות (lower case) בלבד.

**יש לשים לב:** למילים בקוד המכונה הנוצרות משפט הנחיה לא מצורף השדה E, A, R, והערך המוגדר על ידי הנחיה מלא את כל 15 הסיבות של המילה.

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

#### 1. הנחיה '.data'

הפרמטרים של הנחיה '.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 למספר יכולים להופיע רווחים וטאבים בכל מקומות (או בכלל לא), אולם הפסיק חייב להופיע בין המספרים. כמו כן, אסור שיופיע יותר מפסק אחד בין שני מספרים, וגם לא פסק אחרி המספר האחרון או לפני המספר הראשון.

המשפט '.data.' מנהה את האסמבול להקצות מקום בתמונות הנתונים (data image), אשר בו יאחסנו הערךים של הפרמטרים, ולקדם את מונה הנתונים, בהתאם למספר הערכים. אם בהנחתה .data. מוגדרת תווית, אז תווית זו מקבלת את ערך מונה הנתונים (לפני הקידום), ומוכנסת אל טבלת הסמלים. דבר זה מאפשר להתייחס אל מקום מסוים בתמונות הנתונים דרך שם התווית (למעשה, זהה דרך להגדיר שם של משתנה).

כלומר אם נכתב :

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

אז יוקצו בתמונות הנתונים ארבע מיללים רצופות שיכילו את המספרים שמופיעים בהנחיה. התווית XYZ מזוהה עם כתובות המילה הראשונה.

אם נכתב בתכנית את ההוראה :  
mov XYZ, r1

אזי בזמן ריצת התכנית יוכנס לאוגר 1ז הערך 7.

וAILו ההוראה :

lea XYZ, r1

תכנס לאוגר 1ז את ערך התווית XYZ (כלומר הכתובת בזיכרון בה מאוחSEN הערך 7).

2. הנקיה '.string'

הנקיה '.string' פרט אחד, שהוא מחרוזת חוקית. תווים המחרוזת מקודדים לפי ערכי ASCII המאימים, ומוכנסים אל תומנות הנתונים לפי סדרם, כל TWO בamilah נפרדת. בסוף המחרוזת יתווסף התן '0' (הערך המספרי 0), המסמך את סוף המחרוזת. מונה הנתונים של האסמלבל IKODIM בהתאם לערך המחרוזת (בתוספת מקום אחד עבור התו המסיימ). אם בשורת הנקיה מוגדרת תווית, אזי תווית זו מקבלת את ערך מונה הנתונים (לפניהם קידום) ומוכנסת אל טבלת הסמלים, בדומה כפי שנעשה עבור '.data'. (כלומר ערך התווית יהיה הכתובת בזיכרון שבה מתחילה המחרוזת).

לדוגמא, הנקיה :

STR: .string "abcdef"

מקצת בתומנות הנתונים רץ של 7 מיללים, ומאתחלת את המיללים לקודי ASCII של התווים לפי הסדר במחרוזת, ולאחריהם הערך 0 לסימון סוף מחרוזת. התווית STR מזוהה עם כתובות התחלת המחרוזת.

3. הנקיה '.entry'

הנקיה '.entry' פרט אחד, והוא שם של תווית המוגדרת בקובץ המקור הנוכחי (כלומר תווית שמקבלת את ערכיה בקובץ זה). מטרת הנקיה entry היא לאפיין את התווית זו באופן שיאפשר לקוד אסמלבי הנמצא בקבצי מקור אחרים להשתמש בה (אופרנד של הוראה).

לדוגמא, השורות :

HELLO: .entry HELLO  
add #1,r1

מודיעות לאסמלבל שאפשר להתייחס לקובץ אחר לתווית HELLO המוגדרת בקובץ הנוכחי.

لتשומת לב : תווית המוגדרת בתחלת שורת entry. הינה חסרת משמעות והאסמלבל מתעלם מהתווית זו (אפשר שהאסמלבל יוציא הודעה אזהרה).

4. הנקיה '.extern'

הנקיה '.extern' פרט אחד, והוא שם של תווית שאינה מוגדרת בקובץ המקור הנוכחי. מטרת ההוראה היא להודיע לאסמלבל כי התווית מוגדרת בקובץ מקור אחר, וכי קוד האסמלבי בקובץ הנוכחי עושה בתווית שימוש.

נשים לב כי הנקיה זו תואמת להנקיה 'entry', המופיע בקובץ בו מוגדרת התווית. בשלב הקישור תבוצע התאמה בין ערך התוויות, כפי שנקבע בקוד המוכנה של הקובץ שהגדיר את התווית, בין קידוד ההוראות המשמשות בתווית בקבצים אחרים (שלב הקישור אינו רלוונטי למימוש זה).

לדוגמא, משפט הנקיה '.extern', התואם למשפט הנקיה 'entry', מהדוגמא הקודמת יהיה :

.extern HELLO

**لتשומת לב :** תווית המוגדרת בתחילת שורת `extern`. הינה חסרת משמעות והאSEMBLER מTELט מתוויות זו (אפשר שהאSEMBLER יוציא הודעה אחרת).

#### משפט הוראה :

משפט הוראה מורכב מהחלקים הבאים:

1. תווית אופצינלית.
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 האSEMBLER.

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

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

כאשר יש שני אופרנדים, האופרנדים מופרדים זה מזה ב' , (פסיק). בדומה להנחיה 'data' , לא חייב להיות הצמדה של האופרנדים לפסיק או לשם הפעולה באופן כלשהו. כל כמות של רווחים ו/או טאים בין האופרנדים לפסיק, או בין שם הפעולה לאופרנד הראשון, היא חוקית.

למשפט הוראה עם שני אופרנדים המבנה הבא:

label(optional): opcode source-operand, target-operand

לדוגמה:

HELLO: add r7, B

למשפט הוראה עם אופרנד אחד המבנה הבא:

label(optional): opcode target-operand

לדוגמה:

HELLO: bne XYZ

למשפט הוראה ללא אופרנדים המבנה הבא:

label(optional): opcode

לדוגמה:

END: stop

#### אפיון השדות במשפטים של שפת האSEMBLER

##### תווית :

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

**הגדרה של תווית מסוימת ב' :** (נקודותים). זו זה אינו מהו זה חלק מהתוית, אלא רק סימן המציין את סוף ההגדרה.

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

לדוגמה, התוויות המוגדרות להלן הן תוויות חוקיות.

hEllo:

x

He78902:

**لتשומת לב:** מיללים שמורות של שפת האסםביי (כלומר שם של פעולה או הנחיה, או שם של אונגר) אינן יכולות לשמש גם בשם של תווית.

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

**מספר:**

מספר חוקי מתחילה בסימן אופציוני: ‘או +’, ולאחריו סדרה של ספרות בסיסים עשרוניים. לדוגמה: 5,76 +123, ו匿יה במספרים חוקיים. אין תמייה בשפת אסםביי ביצוג בסיסים אחרים מאשר עשרוני, וניין חמיצה במספרים שאיניהם שלמים.

## מחירות :

מחירות חוקית היא סדרת תו ASCII נראים (שניתנים להדפסה), המוקפים במרקאות כפולה (המרקאות אינן נחשבות חלק מהמחרוזת). דוגמה למחירות חוקית: "hello world".

#### פקיד השדה A,R,E בקורס המבונה

בכל מילה בקוד המוכנה של **הוראה** (לא של נתונם), האסמלבר מכניס מידע עבור תהליכי הקישור והטיעינה. זהו השדה A,R,E (שלוש הסיביות הימניות 2,1,0 בהתאמה). המידע ישמש לתיקונים בקוד בכל פעם שייתעורר לצורך הרצה. האסמלבר בונה מילכתילה קוד שמיועד לטיעינה החל מכתובות 100. התיקונים יאפשרו לטעון את הקוד בכל פעם למקום אחר, בלי צורך לחזור על תהליכי האסמלבי.

בכל מילה של ההוראה, בדיק אחות משלש הסיביות של השדה E, A, R, מכילה 1, ושתי הסיביות הנקראות מאופסות. מפרט שיטות המיענו שהווג קודם מצין איזו סיבית תיכיל 1 בכל שיטת מעון.

**סיבית 'A'** (קיצור של Absolute) באח לציר שתוכנן המלאה אינו תלוי במקום בו זיכרוו בו יייטן בפועל קוד המכוונה של התכנית בעת ביצועה (למשל מלאה המכילה אופרנד מידי).

סיבית 'R' (קיצור של Relocatable) באח לצין שתוכן המילה תלוי במקומם בזיכרון בו יתבצע בפועל קוד המכוון של התוכנית בעת ביצועה (למשל מילה המכילה כתובת של תוכית המוגדרת בקובץ המקור הנוכחי).

סיבית 'E' (קיצור של External) באח לציוו שתוקן המילה תלוי בערכו של סמל חיצוני (External) (למשל מילה המכילה כתובות של תוכיות שמוגדרת בקובץ מקור אחר).

אסמבלר עם שני מעברים

כאשר מקבל האסםברל תכניות בשפט אסםברלי, עליו לעבור על התכנית פעמיים. במעבר הראשון, לשזהות את הסמלים (תוויות) המופיעים בתוכנית, ולתת לכל סמל ערך מסווני שהוא המען בזיכרונו שהסמל מייצג. במעבר השני, באמצעות ערכי הסמלים, וכן קוד-הפעולה ומספריו האוגרים, בונים את קוד המכמה.

לדוגמה : האסמבילר מקבל את התוכנית הבאה בשפת אסמבילר :

```

MAIN:    add   r3, LIST
LOOP:    prn   #48
          lea    STR, r6
          inc   r6
          mov   *r6,K
          sub   r1, r4
          cmp   r3, #-6
          bne   END
          dec   K
          jmp   LOOP
END:     stop
STR:     .string "abcd"
LIST:    .data  6, -9
          .data -100
K:       .data  31

```

קוד המכוונה של התוכנית (הוראות ונתונים) נבנה כך שיתאים לטעינה בזיכרון החל ממין 100 (עשורוני).  
התרגום של תוכנית הדוגמה לקוד ביארי מוצג להלן :

Decimal Address	Source Code	Explanation	Binary Machine Code
0100	MAIN: add r3, LIST	First word of instruction Source register 3 Address of label LIST	001010000010100 000000011000100 000010000010010
0101			
0102			
0103	LOOP: prn #48	Immediate value 48	1100000000000100 000000110000100
0104			
0105	lea STR, r6	Address of label STR Target register 6	010000101000100 00000111101010 0000000000110100
0106			
0107			
0108	inc r6	Target register 6	011100001000100 0000000000110100
0109			
0110	mov *r6,K	Source register 6 Address of label K	000001000010100 000000110000100 000010000101010
0111			
0112			
0113	sub r1, r4	Source register 1 and target register 4	001110001000100 000000001100100
0114			
0115	cmp r3, #-6	Source register 3 Immediate value -6	000110000001100 0000000011000100 111111111010100
0116			
0117			
0118	bne END	Address of label END	101000000010100 000001111001010
0119			
0120	dec K	Address of label K	011100000010100 000010000001010
0121			
0122	jmp LOOP	Address of label LOOP	100100000010100 00000110011010
0123			
0124	END: stop		111100000000100
0125	STR: .string "abcd"	Ascii code 'a'	000000001100001
0126		Ascii code 'b'	000000001100010
0127		Ascii code 'c'	000000001100011
0128		Ascii code 'd'	000000001100100
0129		Ascii code '\0' (end of string)	0000000000000000
0130	LIST: .data 6, -9	Integer 6	0000000000000110
0131		Integer -9	11111111110111
0132	.data -100	Integer -100	111111110011100
0133	K: .data 31	Integer 31	000000000011111

האסמבלר מוחזק טבלה שבה רשומים כל שמות הפעולה של הhorאות 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 המירה לבינארי של אופרנדים שהם מענים סמליים (תוויות), יש צורך לבנות טבלה דומה. אולם בהבדל מהקודים של הפעולות, הידיעים מראש, הרי המענים בזיכרונו עבר הסמלים בשימוש התכנית אינם ידועים, עד אשר תוכנית המקור נסקרה כולה ומגלו כל הגדרות הסמלים.

למשל, בקוד לעיל, האסמבלר אינו יכול לדעת שהסמל END אמרו להיות משוייך לערך 124 (עשורי), והסמל K אמרו להיות משוייך לערך 133, אלא רק לאחר שנקראו כל שורות התכנית.

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

במעבר הראשון נבנית טבלת סמלים בזיכרונו, ובה לכל סמל שבתוכנית המקור משוייך ערך מסווני, שהוא מען בזיכרונו. בדוגמה לעיל, טבלת הסמלים לאחר מעבר ראשון היא:

סמל	ערך (בסיס עשרוני)
MAIN	100
LOOP	103
END	124
STR	125
LIST	130
K	133

במעבר השני נעשית המירה של קוד המקור לקוד מכונה. בתחלת המעבר השני צריכים הערכיהם של הסמלים להיות כבר ידועים.

لتשומת לב: תפקיד האסמבלר, על שני המבערים שלו, לתרגם קוד מקור לקוד בשפת מכוונה. בغمר פעולת האסמבלר, התכנית טרם מוכנה לטעינה לזכרונו לצורך ביצוע. קוד המכונה חייב לעבור לשבי הקישור/טעינה, ורק לאחר מכן לשבל הביצוע (שלבים אלה אינם חלק מהממ"ז).

### המעבר הראשון

במעבר הראשון נדרשים כללים כדי לקבוע איזה מען ישוייך לכל סמל. העיקרונו הבסיסי הוא לספור את המוקומות בזיכרון, אותן תופסות הhorאות. אם כל הוראה תיעש בזיכרון למקום העוקב להוראה הקודמת, תציין ספרירה זאת את מען הhorאה הבאה. הספרירה נעשית על ידי האסמבלר ומווחקת במנוע הhorאות (IC). ערכו ההתחלתי של IC הוא 100 (עשורי), ולכן המוכונה של הhorאה הראשונה נבנה כך שייטען לזכרונו החל ממען 100. ה-IC מתעדכן בכל שורת הוראה המכזיקה מקום בזיכרון. לאחר שהasmblar קבע מהו אורך הhorאה, ה-IC מוגדל במספר התאים (밀ילים) הנתפסים על ידי הhorאה, וכך הוא מביע על התא הפניו הבא.

כאמור, כדי לקודד את הhorאות בשפת מכוונה, מוחזק האסמבלר טבלה, שיש בה קוד מתאים לכל שם פעולה. בזמן התרגום מחליף האסמבלר כל שם פעולה בקוד שלה, וכל אופרנד בקידוד מתאים. אך פעולת החלפה אינה כה פשוטה. הhorאות משתמשות בשיטות מיון מגוונות לאופרנדים. אחת פעללה יכולה לקבל ממשמעות שונות, בכל אחת ממשיות המיון, ולכן ניתן לה קידודים שונים לפי שיטות המיון. לדוגמה, פעולת החזזה שומר יכולת להעתיקת תוכן תא זיכרון לאוגן, או להעתיקת תוכן אוגן לאחרר, וכן הלאה. ככל אפשרות זאת של שומר עשוי להתאים קידוד שונה.

על האסמבלר לסרוק את שורת הhorאה בשולמותה, ולהחליט לגבי הקידוד לפי האופרנדים. בדרך כלל מתחלק הקידוד לשדה של שם הפעולה, ושדות נוספים המכילים מידע לגבי שיטות המיון.

הערת: זה לא בהכרח כך בכל מחשב. יש מחשבים בהם, למשל, כל הפוקודות הן עלות אופרנד יחיד (וחפניות מותבצעות על אופרנד זה ועל אוגר קבוע). יש גם מחשבים עם פוקודות של שלשה אופרנדים (כאשר האופרנד השלישי משמש לאחסון תוצאה הפעולה), ועוד אפשרויות אחרות.

כאשר נתקל האסטמבלר בתווית המופיעיה בתחילת השורה, הוא יודע שלפניו הגדרה של תווית, ואז הוא משיק לה מען – תוכנו הנוכחי של IC. כך מתקבלות כל התוויות את מענהן בעות ההגדירה. תוויות אלה מוכנסות לטבלת הסמלים, המכילה בנוסף לשם התווית גם את המען ומאפיינים נוספים. כאשר תהיה התייחסות לתווית באופרנד של הוראה כלשהי, יוכל האסטמבלר לשולף את המען המתאים מטבלת הסמלים.

הוראה יכולה להתיחס גם למספר שורות הטעינה עד כה בתכנית, אלא יוגדר רק בהמשך התכנית. להלן דוגמה, הוראת הסטראוף מען שמוגדר על ידי התווית A שמוינעת רק בהמשך הקוד:

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

כאשר מגיע האסטמבלר לשורת הטעינה (bne A), הוא טרם נתקל בהגדרת התווית A וכמוון לא נתן לה מען, ולכן איןנו יכול להחליף את הסמל A (האופרנד של ההוראה אונ b) במענו בזיכרו. נראה בהמשך כיצד נפתרת בעיה זו.

בכל מקרה, תמיד אפשר לבנות בעבר הראשוני את הקוד הבינארי המלא של המילה הראשונה של כל הוראה, וכן את הקוד הבינארי של כל הנתונים (המתקבלים מההנחיות .string,.data, .).

## המעבר השני

ראינו שבמעבר הראשון, האסטמבלר אינו יכול לבנות את קוד המcona של אופרנדים המשתמשים בסמלים שעדיין לא הוגדרו. רק לאחר שהאסטמבלר עבר על כל התכנית, כך שכל הסמלים נקבעו כבר לטבלת הסמלים, יוכל האסטמבלר להשלים את קוד המcona של כל האופרנדים.

לשום כך עבר האסטמבלר שנית על כל קובץ המקור, וمعدכן את קוד המcona של האופרנדים המשתמשים בסמלים, באמצעות ערכי הסמלים מטבלת הסמלים. זהו המעבר השני, ובסיומו תהיה התוכנית מתורגמת בשלמותה לקוד מcona.

## הפרדת הוראות ונתונים

בתכנית מבחנים בשני סוגים של תוכן: הוראות ונתונים. יש לארכן את קוד המcona 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 הטමונות באירוע הפרדת ההוראות מנתונים היא, שלפעמים עלול המעבד, בעקבות שגיאה לוגית בתכנית, לנסות "לבצע" את הנתונים כאילו היו הוראות חוקיות. למשל, שגיאה שיכולה לגרום תופעה כזו הסטראופת לא נcona. התכנית כMOV לא תעבור כמובן, אך לרוב הנזק הוא יותר חמור, כי נוצרת חריגת חומרה ברגע שהמעבד מבצע פעולה שאינה חוקית.

האסטמבלר שלנו חייב להפריד, בקוד המcona שהוא מייצר, בין קטע הנתונים לבין ההוראות. כלומר בקובץ הפלט (בקוד המcona) תהיה הפרדה של הוראות ונתונים לשני קטיעים נפרדים, ואילו בקובץ הקלט אין חובה שתהייה הפרדה בזו. בהמשך מתואר אלגוריתם של האסטמבלר, ובו פרטיים כיצד לבצע את ההפרדה.

## גילוי שגיאות בתכנית המקור

האסטמבלר אמור לגלות ולדווח על שגיאות בתחריב של תכנית המקור, כגון פעולה שאינה קיימת, מספר אופרנדים שני, סוג אופרנד שאינו מתאים לפעולה, שם אונר לא קיים, ועוד שגיאות אחרות. כמו כן מודא האסטמבלר שככל סמל מוגדר פעם אחת בדיק.

מכאן, שככל שגיאה המתגללה על ידי האסטמבלר נגרמת (בדרכן כלל) על ידי שורת קלט מסוימת. לדוגמה, אם מופיעים שני אופרנדים בהוראה שאמור להיות בה רק אופרנד אחד, האסטמבלר ייתן הודעת שגיאה בנוסח "יותר מדי אופרנדים".

האסטמבלר ידפיס את הודעת השגיאה אל הפלט הסטנדרטי `stdout`. בכל הודעת שגיאה יש לצוין גם את מספר השורה בקובץ המקור בה זוחתה השגיאה.

**لتשומת לב:** האסטמבלר אינו עוצר את פעולה אחריו שנמצאה השגיאה הראשונה, אלא ממשיך לעבר 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).

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

שיטות מיעון חוקיות עבור אופרנד היעד	שיטות מיעון חוקיות עבור אופרנד 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	lea	4
1,2,3	אין אופרנד מקור	clr	5
1,2,3	אין אופרנד מקור	not	6
1,2,3	אין אופרנד מקור	inc	7
1,2,3	אין אופרנד מקור	dec	8
1,2	אין אופרנד מקור	jmp	9
1,2	אין אופרנד מקור	bne	10
1,2,3	אין אופרנד מקור	red	11
0,1,2,3	אין אופרנד מקור	prn	12
1,2	אין אופרנד מקור	jsr	13
אין אופרנד יעד	אין אופרנד מקור	rts	14
אין אופרנד יעד	אין אופרנד מקור	stop	15

## תהליכי העבודה של האסטמבלר

נתאר כתע את אופן העבודה של האסטמבלר. בהמשך, יוצג אלגוריתם שלדי למעבר ראשון ושני.

האסטמבלר מתחזק שני מערכים, שייקראו להלן תMOVות ההוראות (code) וMOVות הנתונים (data). מערכים אלו נתונים למשה תMOVה של זיכרון המcona (כל איבר במערך הוא בגודל מילה של המcona, כולל 15 סיביות). במערך ההוראות בונה האסטמבלר את הקידוד של הוראות המcona שנקרווא במחזור המעבר על קובץ המקור. במערך הנתונים מכינס האסטמבלר את קידוד הנתונים שנקרווא מקובץ המקור (שורות הנקראים מסוג '`.data`' ו- '`.string`').

האסטמבלר משתמש בשני מונחים, שנקראים IC (MOVה ההוראות - Instruction-Counter) ו- DC-1 (MOVה הנתונים - Data-Counter). מונחים אלו מצביעים על המיקום הבא הפניו במערך ההוראות

ובמערך הנתונים, בהתאם. בכל פעם כמשמעות האסטמבלר עובר על קובץ מקור, שני המונחים מאופסים.

בנוספ', מותחן האסטמבלר טבלה, אשר בה נאשפות כל התוויות בהן נתקל האסטמבלר במהלך המעבר על קובץ המקור. טבלה זו קוראים טבלת-הסמלים (symbol-table). לכל סמל נשמרם בטבלה שם הסמל, ערכו המספרי, ומאפיינים שונים, כגון המיקום (data) או סוג הסמל (entry או external).

בעבר הראשו האסטמבלר בונה את טבלת הסמלים ואת השדר של תMOVת הזיכרון (הוראות ונתונים)

האסטמבלר קורא את קובץ המקור שורה אחר שורה, ופועל בהתאם לסוג השורה (הוראה, הנחיה, או שורה ריקה/הערה).

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

2. שורת הווראה :

האסטמבלר מנתח את השורה ומפענח מהי ההווראה, ומהן שיטות המיעון של האופרנדים. מספר האופרנדים נקבע בהתאם להווראה שנמצאה. שיטות המיעון נקבעות בהתאם לתחביר של כל אופרנד, כפי שהושבר לעיל בפרט שיטות המיעון. למשל, התו '#' מצין מיעון מיידי, תווית מצינית מיעון ישיר, שם של אוגר מצין מיעון אוגר ישיר, וכו'.

אם האסטמבלר מוצא בשורת ההווראה גם הגדרה של תווית, אז התווית מוכנסת אל טבלת הסמלים. ערך התווית הוא IC+100, והמאפיין הוא code.

הערה : הערך IC+100 נקבע כדי שקוד המכונה של התכנית יתאים לטעינה לזכרון (לצורך ריצה) החל מכתובת 100.

כעת האסטמבלר קובע לכל אופרנד את ערכו באופן הבא :

- אם זה אוגר – האופרנד הוא מספר האוגר.
- אם זו תווית (מייעון ישיר) – האופרנד הוא ערך התווית כפי שמופיע בטבלת הסמלים (ייתכן והסמל טרם נמצא בטבלת הסמלים, במידה והוא יוגדר רק בהמשך התכנית).
- אם זה התו '#' ואחריו מספר (מייעון מיידי) – האופרנד הוא המספר עצמו.
- אם זו שיטה מיעון אחרת – ערכו של האופרנד נקבע לפי המפרט של שיטת המיעון (ראו תאור שיטות המיעון לעיל).

האסטמבלר מכניס למערך ההווראות, בKİNESSה עליה מציבע מונח ההווראות IC, את קוד המילה הריאונה של ההווראה (בפורמט קידוד כפי שתואר קודם). מילה זו מכילה את קוד הפעולה, ואת מספרי שיטות המיעון. ה- IC מקודם ב-1.

נזכר שכאשר יש רק אופרנד אחד (כלומר אין אופרנד מקור), הסיבות של שיטת המיעון של אופרנד המקור יכילה תמייד 0. בדומה, אם זה הוראה ללא אופרנדים (stop, its), אז הסיבות של שיטות המיעון של שני האופרנדים יכילה 0.

אם זה הוראה בעלת אופרנדים (אחד או שניים), האסטמבלר "משרין" מקום במערך ההווראות עבורAMILות-המידע הנוספות הנדרשות בהוראה זו, ומגדם את IC בהתאם. כאשר אחד או שני האופרנדים הם בשיטה מיעון אוגר-ישיר, אוגר-עקיפן, או מיידי, האסטמבלר מקובד גם את המילויים הנוספות הרלוונטיות במערך ההווראות. מילת-המידע של שיטת מיעון ישיר תישאר ללא קידוד.

3. שורת הנחיה :

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

## I. 'data'.

האסמבלר קורא את רישומי המספרים, המופיעה לאחר 'data', מכניס כל מספר אל מערך הנתונים, ומקדם את מצביע הנתונים DC ב-1 עבור כל מספר שהוכנס.

אם בשורה 'data' מוגדרת גם תווית, אז התווית מוכנסת לטבלת הסמלים. ערך התווית הוא ערך מונה הנתונים DC שלפניהם הכנסת המספרים למערך. המאפיין של התווית הוא 'data'.

## II. 'string'.

הטיפול ב-'string' דומה ל-'data', אלא שקובץ ascii של התווים הם אלו המוכנסים אל מערך הנתונים (כל TWO במילה נפרדת). לבסוף מוכנס למערך הנתונים הערך 0 (המציע סוף-מחוזות). המונה DC מוקדם באורך המחרוזת + 1 (גם התו המסיים את המחרוזת יופס מקום).

הטיפול בתווית המוגדרת בהנחיה 'string'. זהה לטיפול הנעשה בהנחיה 'data'.

## III. 'entry'.

זהה הנחיה לאסמבלר לאפנין את התווית הנתונה כאופrnd entry בטבלת הסמלים. בעת הפתק קבצי הפלט (ראו בהמשך), התווית תירשם בקובץ ה-entries. לתשומת לב: זה לא נחשב כשייניה אם בקובץ המקור מופיעה יותר מהנחיה entry. אחת עם אותה תווית כאופrnd. המופעים הנוספים אינם מושפעים דבר, אך גם אינם מפריעים.

## IV. 'extern'.

זהה הצהרה על סמל (תווית) המוגדר בקובץ מקור אחר, והקובץ הנוכחי עושה בו שימוש. האסמבלר מכניס את הסמל המופיע כאופrnd לטבלת הסמלים, עם הערך 0 (הערך האמתי לא ידוע, וייקבע רק בשלב הקישור), ועם המאפיין external. לא ידוע באיזה קובץ נמצאת הגדרת הסמל, ואין זה רלוונטי עבור האסמבלר. לתשומת לב: זה לא נחשב כשייניה אם בקובץ המקור מופיעה יותר מהנחיה extern. אחת עם אותה תווית כאופrnd. המופעים הנוספים אינם מושפעים דבר, אך גם אינם מפריעים.

יש לשים לב: באופrnd של הוראה או של הנחיה entry, מותר להשתמש בסמל אשר יוגדר בהמשך הקובץ (אם באופן ישיר על ידי הגדרת תווית, ואם באופן עקיף על ידי הנחיה extern.).

בסוף המעבר הראשון, האסמבלר מעדכן בטבלת הסמלים כל סמל המופיע כ-'data', על ידי הוספת (100) + IC (עשורוני) לערכו של הסמל. הסיבה לכך היא שבתמונה הכוללת של קוד המכונה, תמיון הנתונים מופרדת מהתמונות הhorאות, וכל הנתונים נדרשים להופיע בקוד המכונה אחרי כל הhorאות. סמל מסווג data הוא תווית בתמונה הנתונים, והעדכון מוסיף ערך הסמל (כלומר לכתובתו בזיכרון) את האורך הכלול של תמונה הhorאות, בתוספת כתובות התחלה הטיעינה של הקוד, שהיא 100.

טבלת הסמלים מכילה כתעת כל הערכים הנוחים להשלמת תמונה הזיכרון (למעט ערכים של סמלים חיצוניים).

במעבר השני, האסמבלר משלים באמצעות טבלת הסמלים את כל קידוד המילים במערך הhorאות שטרם קודזו במעבר הראשון. במודל המכונה שלנו אלו הן מילוט-מידע נוספת של פקודות, אשר מקודדות אופrnd בשיטת מיון ישיר. האופrnd הוא סמל שמוגדר כפנימי או כחיצוני, וכן בשדה ה-A,R,E הסיבית R או הסיבית E, בהתאם, תהיה 1 (ראו גם מפרט שיטות המיון לעיל).

## אלגוריתם שלדי של האסמבלר

לקידוד ההבנה של תהליך העבודה של האסמבלר, נציג להלן אלגוריתם שלדי למעבר הראשון ולמעבר השני.

لتשומת לב: אין חובה להשתמש דווקא באלגוריתם זה.

כאמור, אנו מחלקים את תמונה קוד המכונה לשני חלקים: תמונה הhorאות (code), ותמונה הנתונים (data). לכל חלק נתחזק מונה נפרד: IC (מונה הhorאות) ו-DC (מונה הנתונים).

כמו כן, נסמן ב- L את המספר הכלול של מילים שתופס קוד המכונה של הוראה נתונה.

#### בנייה את קוד המכונה כך שיתאים לטעינה לזיכרון החל מכתובת 100.

בכל מעבר מתחילה לקרוא את קובץ המקור מההתחלתה.

#### מעבר ראשוני

1. אתחול 0,  $IC \leftarrow 0$ ,  $DC \leftarrow 0$ .
2. קרא את השורה הבאה מקובץ המקור. אם גמר קובץ המקור, עברו ל-16.
3. האם השדה הראשון בשורה הוא סמל? אם לא, עברו ל-5.
4. הדלק דגל "יש הגדרת סמל".
5. האם זהוי הvariable לאחיזון נתוניים, לומר, האם הvariable הוא `string`? אם לא, עברו ל-8.
6. אם יש הגדרת סמל (תוויות), הכנס אותו לטבלת הסמלים עם המאפיין `data`. ערך הסמל יהיה `DC`. (אם הסמל כבר נמצא בטבלה יש להודיע על שגיאה).
7. זזה את סוג הנתוניים, קודד אותו בתמונת הנתוניים, ועדכן את מונה הנתוניים  $DC$  בהתאם לאורך הנתוניים. חזרו ל-2.
8. האם זו הvariable `extern`. או הvariable `entry`? אם לא, עברו ל-11.
9. אם זהוי הvariable `entry`. חזרו ל-2 (ההנחה תופל במעבר השני).
10. אם זו הvariable `extern`. הכנס כל סמל (אחד או יותר) המופיע כאופרנד של ההנחה לתוך טבלת הסמלים, ללא ערך, עם המאפיין `external`. חזרו ל-2.
11. זיהוי שורת הוראה. אם יש הגדרת סמל, הכנס אותו לטבלת הסמלים עם המאפיין `code`. ערכו של הסמל יהיה  $IC + 100$  (אם הסמל כבר נמצא בטבלה יש להודיע על שגיאה).
12. חפש את שם הפעולה בטבלת שמות הפעולות, ואם לא נמצא, או הודיע על שגיאה בשם ההוראה.
13. נחה את מבנה האופרנדים של ההוראה וחשב את L (מספר המילים שתופסת ההוראה בקוד הבינארי).
14. בנה כעת את הקוד הבינארי של המילה הראשונה של ההוראה, ושל כל מילת- מידע נוספת המוקודדת אופרנד שאינו מכיל סמל (ミューון מיידי, מיעון אוגר ישר, מיעון אוגר עקיף).
15. עדכן  $IC \leftarrow IC + L$ , וחזרו ל-2.
16. קובץ המקור נקרא בשלמותו. אם נמצאו שגיאות במעבר הראשון, עצור כאן.
17. עדכן בטבלת הסמלים את ערכו של כל סמל המופיעין כ- `data`, ע"י הוספת הערך  $IC + 100$  (ראה הסבר בהמשך).
18. התחל מעבר שני.

#### מעבר שני

1. אתחול 0,  $IC \leftarrow 0$ .
2. קרא את השורה הבאה מקובץ המקור. אם גמר קובץ המקור, עברו ל-9.
3. אם השדה הראשון בשורה הוא סמל (תוויות), דגל עליו.
4. האם זהוי הvariable. או `string`. או `extern`? אם כן, חזרו ל-2.
5. האם זהוי הvariable ? אם לא, עברו ל-7.
6. הוסף בטבלת הסמלים את המאפיין `entry` לכל סמל (אחד או יותר) המופיע כאופרנד של ההנחה (אם הסמל לא נמצא בטבלת הסמלים, יש להודיע על שגיאה). חזרו ל-2.
7. השלים את הקידוד הבינארי של מילות- המידע של האופרנדים, בהתאם לשיטות המיעון בשימוש. אם אופרנד בקוד המקור מכיל סמל, מצא את ערכו בטבלת הסמלים (אם הסמל לא נמצא בטבלה, יש להודיע על שגיאה).
8. עדכן  $IC \leftarrow IC + L$ , וחזרו ל-2.
9. קובץ המקור נקרא בשלמותו. אם נמצאו שגיאות במעבר השני, עצור כאן.
10. בנה את קבצי הפלט (פרטים נוספים בהמשך).

נפעיל אלגוריתם זה על תוכנית הדוגמה שראינו קודם, ונציג את הקוד הבינארי שמתתקבל במעבר ראשון ובמעבר שני.

```

MAIN:    add   r3, LIST
LOOP:    prn   #48
          lea   STR, r6
          inc   r6
          mov   *r6,K
          sub   r1, r4
          cmp   r3, #-6
          bne   END
          dec   K
          jmp   LOOP
END:     stop
STR:     .string "abcd"
LIST:    .data  6, -9
          .data -100
K:       .data  31

```

מבצע מעבר ראשון על הקוד לעיל, ובנייה את טבלת הסמלים. כמו כן, נשלים במעבר זה את הקידוד של כל תMOVNT הנתונים, ושל המילה הראשונה של כל הוראה. כמו כן, נקודד מילות-מידע וספנות של כל הוראה, ככל קידוד זה אינו תלוי בערך של סמל. את מילות-המידע שעדיין לא ניתן לקודד במעבר הראשון נסמך ב""?" בדוגמה להלן.

Decimal Address	Source Code	Explanation	Binary Machine Code
0100	MAIN: add r3, LIST	First word of instruction	001010000010100
0101		Source register 3	000000011000100
0102		Address of label LIST	?
0103	LOOP: prn #48	Immediate value 48	1100000000000100
0104			000000110000100
0105	lea STR, r6	Address of label STR	010000101000100
0106			?
0107		Target register 6	000000000110100
0108	inc r6		011100001000100
0109		Target register 6	000000000110100
0110	mov *r6,K	Source register 6	000001000010100
0111		Address of label K	000000110000100
0112			?
0113	sub r1, r4	Source register 1 and target register 4	001110001000100
0114			000000001100100
0115	cmp r3, #-6	Source register 3	000110000001100
0116		Immediate value -6	000000011000100
0117			11111111010100
0118	bne END		101000000010100
0119		Address of label END	?
0120	dec K		011100000010100
0121		Address of label K	?
0122	jmp LOOP		100100000010100
0123		Address of label LOOP	?
0124	END: stop		111100000000100
0125	STR: .string "abcd"	Ascii code 'a'	000000001100001
0126		Ascii code 'b'	000000001100010
0127		Ascii code 'c'	000000001100011
0128		Ascii code 'd'	000000001100100
0129		Ascii code '\0' (end of string)	000000000000000
0130	LIST: .data 6, -9	Integer 6	000000000000110
0131		Integer -9	111111111101111
0132	.data -100	Integer -100	111111110011100
0133	K: .data 31	Integer 31	0000000000011111

סמל	ערך (בסיס עשרוני)
MAIN	100
LOOP	103
END	124
STR	125
LIST	130
K	133

נכע עתה את המעבר השני. נשים באמצעות טבלת הסמלים את הקוד החסר במילים המסומנות ???.

הקוד הבינארי בצוותו הוסיף לכך שגם בשלב תחילת הנושא "אסמבולר עם שני מעברים".

הערה : כאמור, האסמבולר בונה קוד מכונה כך שייתאים לטעינה לזכרון החל מכתובת 100 (עשרוני).

אם הטעינה בפועל (לצורך הריצת התכנית) תהיה לכתובת אחרת, יידרשו תיקונים בקוד הבינארי בשלב הטעינה, שיוכנסו בעורת מידע נוסף שהאסמבולר מכין בקבצי הפלט (ראו בהמשך).

Decimal Address	Source Code	Explanation	Binary Machine Code
0100	MAIN: add r3, LIST	First word of instruction	001010000010100
0101		Source register 3	000000011000100
0102		Address of label LIST	000010000010010
0103	LOOP: prn #48		1100000000000100
0104		Immediate value 48	000000110000100
0105	lea STR, r6		010000101000100
0106		Address of label STR	00000111101010
0107		Target register 6	0000000000110100
0108	inc r6		011100001000100
0109		Target register 6	0000000000110100
0110	mov *r6,K		000001000010100
0111		Source register 6	000000110000100
0112		Address of label K	000010000101010
0113	sub r1, r4		001110001000100
0114		Source register 1 and target register 4	0000000001100100
0115	cmp r3, #-6		000110000001100
0116		Source register 3	0000000011000100
0117		Immediate value -6	111111111010100
0118	bne END		1010000000010100
0119		Address of label END	00000111001010
0120	dec K		0111000000010100
0121		Address of label K	000010000001010
0122	jmp LOOP		1001000000010100
0123		Address of label LOOP	000001100111010
0124	END: stop		1111000000000100
0125	STR: .string "abcd"	Ascii code 'a'	0000000001100001
0126		Ascii code 'b'	0000000001100010
0127		Ascii code 'c'	0000000001100011
0128		Ascii code 'd'	0000000001100100
0129		Ascii code '\0' (end of string)	0000000000000000
0130	LIST: .data 6, -9	Integer 6	0000000000000110
0131		Integer -9	111111111110111
0132	.data -100	Integer -100	111111110011100
0133	K: .data 31	Integer 31	0000000000011111

בסוף המעבר השני, אם לא נתגלו שגיאות, האסמבלר בונה את קבצי הפלט (ראו בהמשך), שמכילים את הקוד הבינארי ומידע נוסף שלבי הקישור והטיענה. כאמור, שלבי הקישור והטיענה אינם מיושם בפרויקט זה, ולאណון בהם כאן.

### קבצי קלט ופלט של האסמבלר

בפעולת האסמבלר, יש להעיר אליו באמצעות ארגומנטים של שורת הפקודה (command line arguments) רשימה של שמות קבצי מקור (אחד או יותר). אלו הם קבצי טקסט, ובهم תכניות בתחריר של שפת האסמבלי שהוגדרה במופיע זה.

האסמבלר פועל על כל קובץ מקור בנפרד, ויוצר עבורו קבצי פלט כדלקמן:

- קובץ object, המכיל את קוד המוכונה.
- קובץ externals, שבו פרטים על כל המיקומות (הכתובות) בקוד המוכונה בהם מקודד ערך של סמל חיצוני (סמל שהוגדר באמצעות ההנחייה .extern, ומופיע בטבלת הסמלים - external).
- קובץ entries, שבו פרטים על כל סמל שמוסחර כנקודות כניסה (סמל שהופיע כאופרנד של הנחיה .entry, ומופיע בטבלת הסמלים - entry).

אם אין בקובץ המקור אף הנחיה .extern, האסמבלר לא יוצר את קובץ הפלט מסווג externals. אם אין בקובץ המקור אף הנחיה .entry, האסמבלר לא יוצר את קובץ הפלט מסווג entries.

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

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

```
assembler x y hello
```

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

שמות קבצי הפלט מבוטסים על שם קובץ הקלט, כפי שהופיע בשורת הפקודה, בתוספת סיומה מותאמת : הסיומת ".ob". עברו קובץ ה-object, הסיומת ".ent" עברו קובץ ה-entries, והסיומת ".ext" עברו קובץ ה-externals.

לדוגמא, בהפעלת האסמבלר באמצעות שורת הפקודה : assembler x y hello. יוצר קובץ פלט .ob, וכן קבצי פלט .ext ו-.ent. ככל שיש הנחיות entry. או extern. בקובץ המקור.

נציג כעת את הפורמטים של קבצי הפלט. דוגמאות יובאו בהמשך.

### פורמט קובץ ה- object

קובץ זה מכיל את תמונת הזיכרון של קוד המוכונה, שני חלקים : תמונת ההוראות ראשונה, ואחריה ובצמוד לתמונה הנותנים.

כזכור, האסמבלר מקודד את ההוראות כך שתמונה ההוראות מתאימים לטיענה החל מכתובת 100 (עשרוני) בזיכרון. נשים לב שرك בסוף המעבר הראשון יודיעים מהו הגודל הכלול של תמונת ההוראות. מכיוון שתמונה הנותנים נמצאת אחרי תמונה ההוראות, גודל תמונה ההוראות משפיע על הכתובות בתמונה הנותנים. זו הסיבה שבגללה היה צורך לעדכן בטבלת הסמלים, בסוף המעבר הראשון, את ערכי הסמלים המאופיינים כ-data (כזכור, הוספנו לכל סמל כזה את הערך IC+100). במעבר השני, השלמת הקידוד משתמשת בערכים המעודכנים של הסמלים, המותאמים למבנה המלא והסופי של תמונת הזיכרון.

כעת האסטבלר יכול לכתוב את תמונה זויכרונו בשלהמזהה לתוך קובץ פלט (קובץ ה-object).

השורה הראשונה בקובץ ה-object היא "cotract", המכילה שני מספרים (בבסיס עשרוני) : הראשון הוא האורך הכלול של תמונה ההוראות (ב밀ות זיכרונו), והשני הוא האורך הכלול של תמונה הנטוינס (ב밀ות זיכרונו). בין שני המספרים יש רווח אחד.

השורות הבאות בקובץ מכילות את תמונה הזיכרונו. בכל שורה שני שדות: כתובות של מילה בזיכרונו, ותוקן המילה. הכתבת תירשם בסיסי עשרוניארבע ספרות (כולל אפסים מוביילים). תוכן המילה יירשם בסיס אוקטלי ב-5 ספרות. בין שני השדות יש רווח אחד.

#### פורמט קובץ ה-entries

קובץ ה-entries בנייתו משוראות טקסט, שורה אחת לכל סמל שמאופיין כ-entry. בשורה מופיע שם הסמל, ולאחריו ערכו כפי שנקבע בטבלת הסמלים (בבסיס עשרוני). אין חשיבות לסדר השורות, כי כל שורה עומדת בפני עצמה.

#### פורמט קובץ ה-externals

קובץ ה-externals בנייתו אף הוא משוראות טקסט, שורה לכל כתובות בקוד המכונה בה מקודדת מילת-מידע המתאפיינת בסמל שמאופיין כ-external. בשורה מופיע שם הסמל, ולאחריו הכתובת של מילת-המידע (בבסיס עשרוני). אין חשיבות לסדר השורות, כי כל שורה עומדת בפני עצמה.

כמובן שייתכן ויש מספר כתובות בקוד הקידוד מתיאחס לאותו סמל חיצוני. לכל כתובות צו תהיה שורה נפרדת בקובץ ה-externals.

נדגים את הפלט שמייצר האסטבלר עבור קובץ מקור בשם ps.as הנטו להלן.

; file ps.as

```
.entry LIST
.extern fn1
MAIN:    add    r3, LIST
          jsr    fn1
LOOP:     prn   #48
          lea    STR, r6
          inc    r6
          mov    *r6, L3
          sub    r1, r4
          cmp    r3, #-6
          bne    END
          add    r7, *r6
          clr    K
          sub    L3, L3
.entry MAIN
          jmp    LOOP
END:      stop
STR:      .string "abcd"
LIST:     .data   6, -9
          .data   -100
K:        .data   31
.extern L3
```

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

Decimal Address	Source Code	Explanation	Binary Machine Code
0100	MAIN: add r3, LIST	First word of instruction Source register 3 Address of label LIST	001010000010100 000000011000100 000010001001010
0101			
0102			
0103	jsr fn1		110100000010100 000000000000001
0104		Address of label fn1 (external)	
0105	LOOP: prn #48		1100000000001100 000000110000100
0106		Immediate value 48	
0107	lea STR, r6		010000101000100 000010000100010 000000000110100
0108		Address of label STR	
0109		Target register 6	
0110	inc r6		011100001000100 000000000110100
0111		Target register 6	
0112	mov *r6, L3		000001000010100 000000110000100 000000000000001
0113		Source register 6	
0114		Address of label L3 (external)	
0115	sub r1, r4		001110001000100 000000001100100
0116		Source register 1 and target register 4	
0117	cmp r3, #-6		000110000001100 000000011000100 111111111010100
0118		Source register 3	
0119		Immediate value -6	
0120	bne END		101000000010100 000010000011010
0121		Address of label END	
0122	add r7, *r6		001010000100100 000000111110100
0123		Source register r0 and target register 6	
0124	clr K		010100000010100 000010001100010
0125		Address of label K	
0126	sub L3, L3		001100100010100 000000000000001 000000000000001
0127		Address of label L3 (external)	
0128		Address of label L3 (external)	
0129	jmp LOOP		100100000010100 000001101001010
0130		Address of label LOOP	
0131	END: stop		111100000000100
0132	STR: .string "abcd"	Ascii code 'a'	000000001100001
0133		Ascii code 'b'	000000001100010
0134		Ascii code 'c'	000000001100011
0135		Ascii code 'd'	000000001100100
0136		Ascii code '\0' (end of string)	000000000000000
0137	LIST: .data 6, -9	Integer 6	000000000000110
0138		Integer -9	11111111110111
0139	.data -100	Integer -100	111111110011100
0140	K: .data 31	Integer 31	000000000011111

לחלה תוכן קבצי הפלט של הדוגמה.

:הקובץ ps.ob

32 9  
0100 12024  
0101 00304  
0102 02112  
0103 64024  
0104 00001  
0105 60014  
0106 00604  
0107 20504  
0108 02042  
0109 00064  
0110 34104  
0111 00064  
0112 01024  
0113 00604  
0114 00001  
0115 16104  
0116 00144  
0117 06014  
0118 00304  
0119 77724  
0120 50024  
0121 02032  
0122 12044  
0123 00764  
0124 24024  
0125 02142  
0126 14424  
0127 00001  
0128 00001  
0129 44024  
0130 01512  
0131 74004  
0132 00141  
0133 00142  
0134 00143  
0135 00144  
0136 00000  
0137 00006  
0138 77767  
0139 77634  
0140 00037

:הקובץ ps.ent

:הקובץ ps.ext

MAIN 100	fn1 0104
LIST 137	L3 0114
	L3 0127
	L3 0128

## סיכום והנחיות כלליות

- גודל תוכנית המקור הניתנת כקלט לאסטבלר אינו ידוע מראש, וכך גם גודלו של קוד המוכנה אינו צפוי מראש. אולם בכך להקל בימוש האסטבלר, מותר להניח גודל מקסימלי. לפיכך יש אפשרות להשתמש במערכות לאחסן תMOVת קוד המוכנה בלבד. כל מבנה נתונים אחר (למשל טבלת הסמלים), יש למשול באופן ייעיל וחסוני (למשל באמצעות רשימה מקושרת והקצתה זיכרון דינמי).
- השמות של קבצי הפלט צריים להיות תואמים לשם קובץ הפלט, למעט הסיומות. למשל, אם קובץ הפלט הוא prog.as או קבצי הפלט שיוצרו הם : prog.ob, prog.ext, prog.ent.
- מתכונת הפעלת האסטבלר צריכה להיות כפי הנדרש בממ"ז, ללא שינויים כלשהם. ככלומר, ממשך המשמש יהיה אך ורק באמצעות שורת הפקודה. בפרט, שמות קבצי המקור יועברו לתכנית האסטבלר כארומנטים בשורת הפקודה. אין להוציא פרטיטי קלט אינטראקטיביים, חלונות גרפיים למיניהם, וכו'.
- יש להקפיד לחלק את מימוש האסטבלר במספר מודולים (קובצים בשפט C) לפי MISMOOT. אין לרכז MISMOOT מסווגים שונים במודול יחיד. מומלץ לחלק למודולים כגון : מעבר ראשוני, פונקציות עזר (למשל, תרגום לבסיס, ניתוח תחבירי של שורה), טבלת הסמלים, מפת הזיכרון, טבלאות קבועות (קודם הפעולה, שיטות המיעון החוקיקות לכל פעולה, וכו').
- יש להקפיד ולתעד את המימוש באופן מלא וברור, באמצעות העורות מפורטוות בקוד.
- יש לאפשר תווים לבנים עודפים בקובץ הפלט בשפט אסטבלר. למשל, אם בשורת הוראה יש שני אופרנדים המופרדים בפסיק, או זכי פנוי ואחרי הפסיק מותר שייהיו רווחים וטאבים בכל כמות. בדומה, גם לפני ואחרי שם הפעולה. מותירות גם שורות ריקות. האסטבלר יתעלם מתחומים לבנים מיותרים (כלומר יידלג עליהם).
- הקלט (קוד האסטבלר) עלול להכיל שגיאות תחביריות. על האסטבלר לגולות ולדוח על כל השורות השגויות בקלט. אונלעוצר את הטיוף בקובץ קלט לאחר גילוי השגיאה הראשונה. יש להפסיק למשך הדעות מפורטות ככל הניתן, כדי שאפשר יהיה להבין מה והיכן כל שגיאה. כמוון שם קובץ קלט מכיל שגיאות, אונ טעם להפיק עבورو את קבצי הפלט (ob, ext, ent).

**תס ושולט פרק החסרים והגדרת הפרויקט.**

**בשאלות ניתן לפנות ל专家组 הדיון באתר הקורס, ואל כל אחד מהמנחים בשעות הקבלה שלهما.**

lezicircum, אפשרותו של כל סטודנט לפנות לכל מנהה, לאו דווקא למנחה הקבוצה שלו, לקבלת עזרה. שוב מומלץ לכל אלה שטרם בדקו את התכנים באתר הקורס לעשות זאת. נשאלות באתר זה הרבה שאלות בנושא חומר הלימוד והממיינים, והתשובות יכולות להועיל לכם.

لتשובות לבכם : לא תיתנו דחיה בהגשת הממיין, פרט למקרים מיוחדים כגון מילואים או מחלוקת ממושכת. במקרים אלו יש לבקש ולקבל אישור מראש מצוות הקורס.

**בהצלחה !**