

## מטלת מנהה (ממ"ן) 14

הקורס : 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטלת : פרויקט גמר

משקל המטלת : 31 נקודות

מספר השאלות : 1

מועד אחרון להגשה : 9.8.2015

סמסטר : 2015/2016

### 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"נוהל 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.

עליך לכתב תוכנת אסמבילר, לשפת אסמביל שותוגדר בהמשך. הפרויקט ייכתב בשפת C. עלייך להגיש :

1. קבצי המקור של התוכנית שתכתב (קבצים בעלי סימות `C` או `.c`).
2. קבצי הרצה.
3. הגדרת טבבית העבודה (`MAKEFILE`).
4. דוגמא לקבצי קלט וקבצי הפלט, שנוצרו על ידי הפעלת התוכנית שלך על קבצי קלט אלה.

בשל גודל הפרויקט, עליך לחלק את התוכנית למספר קבצי מקור. יש להקפיד שהקובד הנמצא בתוכניות המקור יעמוד בקריטריונים של בהירות, קריאות וככינה.

זכיר מספר היבטים חשובים :

1. הפיטהה של מבני הנתונים : רצוי (במידת האפשר) להפריד בין הגישה למבנה הנתונים לבין השימושelmanם. כך, למשל, בעת כתיבת שגרות לטיפול במחסנית, אין זה מעוניינים של המשמשים בשגרות אלה, אם המחלשיות ממומשת באמצעות מערך או באמצעות רשימה מקוושת.

2. קריאות הקובד : רצוי להציג על הקבאים הרלוונטיים בנפרד, תוך שימוש בפקודת `#define`, ולהימנע מ"מספרי קסם", שימושם נהירה לך בלבד.

3. תיעוד : יש להכנס בקבצי המקור תיעוד תמציתי וברור, שיסביר ותפרקיה של כל פונקציה ופונקציה. כמו כן יש להסביר את תפקידם של משתנים חשובים.

הערה : תוכנית "עובדת", דהיינו תוכנית שביצעת את הדריש ממנה, אינה עורובה לציון גבוה. כדי לקבל ציון גבוה על התוכנית לעמוד בקריטריונים לעיל, אשר משקלם המשותף מגע עד לכ-40% ממשקל הפרויקט.

הפרויקט כולל כתיבה של תוכנית אסמבילר עבור שפת אסמבלי, שהוגדרה במיוחד עבור פרויקט זה. מותר לעבוד בזוגות. אין לעבוד בצוות גדול יותר מאשר שלושה. **פרויקטיהם שיוגשו בשלישיות או יותר לא יבוצעו.** חובה שניי סטודנטים, הבוחרים להגיש יחד את הפרויקט, יהיו **שייכים לאותה קבוצה.**

מומלץ לקרוא את הגדרת הפרויקט פעם ראשונה ברצף, לקבלת תמונה כללית לגבי הנדרש, ורק לאחר מכן לקרוא בשנית, בצורה מעמיקה יותר.

#### רקע כללי

כידוע, קיימות שפות תכנות רבות, ומספר גדול של תוכניות, הכתובות בשפות שונות, עשויות לרוץ באופןו מחשב עצמו. כיצד "מכיר" המחשב כל כך הרבה התשובה פשוטה: המחשב מכיר למעשה שפה אחת בלבד: הוראות ונתונים המובאים בקוד בינארי. קוד זה מאוחסן בשום זיכרון, ונראה כמו רצף של ספרות בינאריות. יחידת העיבוד המרכזי - היע"מ (CPU) - יודעת לפרק את הרצף הזה לקטעים קטנים בעלי משמעות: הוראות, מענים ונתונים. אופן הפירוק נקבע, באופן חד משמעי, על ידי המיקרו קוד של המעבד.

למעשה, זיכרון המחשב כולל הוא אוסף של סיביות, שנוהגים לראותן כמקובצות לייחדות בעלות אורך קבוע, הנקראות בתים. כאשר נמצאת בזכרון תוכנית משתמש, לא ניתן להבחין, בין שאינה מiomנת, בהבדל פיסי כלשהו, בין אותו חלק בזכרון, שבו נמצאת התוכנית, לבין שאר הזיכרון.

יחידת העיבוד המרכזי (היע"מ) יכולה לבצע מספר מסויים של הוראות פשוטות, ולשם כך היא משתמשת בכמה אוגרים (register), הקיימים בה. **זוגמאות:** העברת מספר מתא בזכרון לאוגר ביע"מ או בחזרה, הוספה 1 למספר הנמצא באוגר, בדיקה האם מספר המאוחסן באוגר שווה לאפס. הוראות פשוטות אלה ושילובים שלתן הן ההוראות המרכיבות את תוכנית המשמשת שהיא נמצאת בזכרון. כל תוכנית כפוי לנכונותה כדי המוכננת, וועבר בסופו של דבר באמצעות תוכנה מיוחדת לצורה סופית זו.

סביר כיצד מתבצע קוד זה: כל הוראה בקוד יכולה להתיחס לנตอน הנמצא בהוראה עצמה, לאוגר או לפחות בזכרון. היע"ם מפרקת כל שורת קוד להוראה ולאופרנדים שלה, ובמקרה של ההוראה. אוגר מיוחד בתוכה היע"ם מצביע תמיד על ההוראה הבאה לביצוע (program counter). כאשר מגיעה היע"ם להוראת עצירה, היא מחזירה את הפיקוד לתוכנית שהפעילה אותה או למערכות ההפעלה.

לכל שפת תכנות יש, כידוע, מחר (compiler), או מפרש (interpreter), המתרגם תוכניות מקור לשפת מוכנה. אם תוכנית מקור נכתבת בשפת אסמבלי, נקראת התוכנית המתרגם בשם אסמבילד. בפרויקט זה עלייך לכתוב אסמבילד. לשם כך你需要 אחראי גלולה של תוכנית שנכתבה בשפת אסמבלי, שנגיד במיוחד עבור פרויקט זה, עד לשילוחה לתוכנת הקישור והטעינה (linker/loader).

**לשומת לב:** בהסבירים הכלליים על אופן העבודה תוכנת האסמבילד, יש לתיחסות גם לעבודות תוכנת הקישור (linker) ותוכנת הטעינה (loader). התיחסויות אילו הובאו על מנת לאפשר לכם להבין את המשך תהליך העיבוד של הפלט של תוכנת האסמבילד. אין לטעות: עלייך לכתוב את תוכנות האסמבילד בלבד, אין צורך לכתוב גם את תוכנת הקישור והטעינה!!!

תחליה נגדיר את שפת האסמבילד ואת המחשב הדמיוני, שהגדנו עבור פרויקט זה.

### "חומרה":

המחשב בפרויקט מורכב מיעם (יחייזת עיבוד מרכזית), אוגרים ו זיכרון RAM, כאשר חלק מהזיכרון משמש גם כמחסנית (stack). גודלה של מלת זיכרון במחשב הוא 12 סיביות. האריתמטיקה נעשית בשיטת המשלים ל-2 (2's complement). מחשב זה מטפל רק במספרים שלמים חיוביים ושליליים, אין טיפול במספרים ממשיים.

### אוגרים:

למחשב 8 אוגרים כלליים (r0,r1,r2,r3,r4,r5,r6,r7) (PC – program counter), מונה תוכנית (SP – stack pointer), מצביע המחסנית של זמן ריצה (PSW - program status word) – בעל שני דגלים: דגל נשא (Carry) ודגל אפס (Zero). גודלו של כל אוגר הוא 12 סיביות.

עבור ה-PSW הסיביות הראשונות הן C ו-Z, כלומר בתחביר של שפת C :

$$C = (PSW \& 01)$$

$$Z = (PSW \& 02)$$

גודל הזיכרון הוא 1000 תאים, וכל תא הוא בגודל של 12 סיביות.

קידוד של תווים (characters) נעשה בקוד ascii.

### מבנה הוראות מכונה:

בל הוראות מכונה מקודדת למספר מילוט זיכרון, החל מ- תא אחד ועד למקSYMOM של 3 תא זיכרון, הכל בהתאם לשיטות המיעון בהן נעשה שימוש. בכל סוג ההוראות, המבנה של המילה הראשונה זהה. מבנה המילה הראשונה בהוראה הוא כדלהלן:

11 10	9 8 7 6	5	4	3	2	1	0
group	opcode	מייען אופרנד יעד	מייען אופרנד מקור			E,R,A	

סיביות 9-6 מהוות את קוד ההוראה (opcode). בשפה שלנו יש 16 קודי הוראה והם :

הקוד בבסיס דצימלי (10)	הקוד בבסיס 4	פעולה
0	0	mov
1	1	cmp
2	2	add
3	3	sub
4	10	not
5	11	clr
6	12	lea
7	13	inc
8	20	dec
9	21	jmp
10	22	bne
11	23	red

12	30	prn
13	31	jsr
14	32	rts
15	33	stop

ההוראות נכתבות תמיד באOTTיות קטנות. פרוט' משמעות ההוראות יבוא בהמשך. מיד לאחר שם ההוראה ובצמוד אליה יופיע המספר 1 או 2 המציין כמה פעמיים יש לקובד את הפקודה, למשל **mov1** יש לקובד פעם אחת ואילו **mov2** קובד פעמיים.

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

. סיביות 3-2 מקודדות את מספרה של שיטת המיעון של אופרנד היעד (destination operand) בשפה שלטן קיימות ארבע שיטות מייעון, שמספרן הוא בין 0 ל- 3.

השימוש בשיטות מייעון, מצריך קובד של מילوت-מידע נוספת. אם שיטת המיעון של רק אחד משני האופרנדים, דורשת מילوت מידע נוספת, אז מילوت המידע הנוספות מתייחסות לאופרנד זה. אך אם שיטת המיעון של שני האופרנדים דורשת מילوت-מידע נוספת, אז מילות-המידע הנוספות הרשונות מתייחסות לאופרנד המקור (source operand) ומילות-המידע הנוספות האחרונות מתייחסות לאופרנד היעד (destination operand).

גם למילות-המידע הנוספות יהיה זוג סיביות בצל ימין, המציינות את השזה E,A,R.

#### סיביות 11-10 (group)

סיביות אלו מסמלות את סוג ההוראה המקודדת. בשפה קיימות הוראות ללא אופרנדים, הוראות עם אופרנד יחיד, והוראות עם 2 אופרנדים. ערכי סיביות אלה יהיו 00 עבור קובד הוראה ללא אופרנדים. 01 עבור קובד הוראה עם אופרנד יחיד, ו- 10 עבור קובד הוראה עם 2 אופרנדים.

#### סיביות 1-0 (E,R,A)

סיביות אלה מראות את סוג הקובד, האם הוא מוחלט (Absolute , חיצוני (External) או מצריך מיקום חדש (Relocatable). ערך של 00 משמעו שהקובד הוא מוחלט. ערך של 01 משמעו שהקובד הוא חיצוני. ערך של 10 משמעו שהקובד מצריך מיקום חדש. סיביות אלה מתווספות רק לקובדים של הוראות, והן מתווספות גם למיללים הנוספות שיש לקובדים אלה.

ארבע שיטות המיעון הקיימות במכונה שלנו הן :

ערך	שיטות מייעון	תוכן המילה נוספת	אופן הכתיבה	דוגמא
0	מייעון מידי	המילה נוספת מכילה את האופrnd מתחילה בטו # ואחריו ובצמוד אליו מופיע מספר חוקי.	האופrnd מתחילה בטו # ואחריו מיצג ב- 10 סיביות אליהם מתייחסות זוג סיביות של שדה A,R,E	mov #-r2
1	מייעון ישיר	האופrnd הינו תווית שהוחזרה או תוצאה בהמשך הקובץ. התחזרה נמשכת על ידי כתיבת תווית בקובץ (בפקודת .data או .string או בגדרתת תווית ליד שורת קוד של התוכנית). או על ידי הניתית .extern	המילה נוספת מכילה מען בזיכרונו. תוכן מען זה הינו האופrnd המבוקש, מיצג ב- 10 סיביות אליה מתייחסות זוג סיביות של שדה A,R,E	mov x, r2
2	מייעון העתק קודם	משמעות שיטת מייעון זו (\$\$) היא שהאופrnd זהה לאופrnd הראשון של הפקודה הקודמת. לכך נבע שלא ניתן להשתמש בשיטת מייעון זו בפקודה הראשונה (אין לה פקודה קודמות) וכן, לא בכל המקרים אפשרי להפעיל שיטת מייעון זו (יש לבדוק מקרים אלה ולהדפיס הודעה שגיאה מתאימה)	האופrnd מורכב מ-	mov \$\$,r2
3	מייעון אוגר ישיר	אם האוגר משמש כאופrnd יעד, הוא יקודד במילה נוספת שתכיל ב-5 הסיביות הימניות את מספרו של האוגר. אם האוגר הוא אופrnd מקור, הוא יקודד במילה נוספת שתכיל ב-5 הסיביות השמאליות את מספרו של האוגר. אם 2 אופrndים הם אוגרים הם יחלקו מילה נוספת משותפת. כאשר 5 הביטים הימניים הם עבור אוגר היעד, ו- 5 הביטים השמאליים הם עבור אוגר המקור. לייצוג זה מתייחסות זוג סיביות של שדה A,R,E	האופrnd הינו שם חוקי של אוגר.	mov r1,r2

הערה: מותר להתניחס לתווית עוד לפני שמצוירים עליה (באופן סתום או מפורש), בתנאי שהיא אכן מוחדרת במקום כלשהו בקובץ.

### אפיון הוראות המכונה:

ההוראות המכונה מתחולקות לשלווש קבוצות, לפי מספר האופרנדים הדרושים להן.

#### קבוצת ראשונה:

ההוראות הזוקקות לשני אופרנדים. הפקודות השויות לקבוצה זו הן:

mov, cmp, add, sub, lea

פקודת	הסבר פעולה	דוגמא	הסבר דוגמא
mov	מבצעת העתקה של האופרנד הראשון, אופרנד המקור (source) אל האופרנד השני (destination) היעד (dest). בהתאם לשיטת המיעוון.	mov A, r1	העתק תוכן משתנה לאויגר r1.
cmp	מבצעת "השוואה" בין שני האופרנדים שלה. אופן ההשוואה: תוכן אופרנד היעד (השני) מופחת מתוכן אופרנד המקור (הראשון), ללא שמיינת תוצאת החישור. פעלת החישור מעדכנת את דגל האפס, דגל Z, באויגר הסטטוס, PSW.	cmp A, r1	אם תוכן הערך הנמצא במעט זהה לתוכנו של אויגר r1 אז דגל האפס, Z, באויגר הסטטוס, PSW יודלק, אחרת הוא יאפס.
add	אופרנד היעד (השני) מקבל את סכום תוכן משתנה סכום אופרנד המקור (הראשון) והיעד (השני).	add A, r0	אויגר r0 מקבל את סכום תוכן משתנה וארכו הנקחי של A אויגר r0.
sub	אופרנד היעד (השני) מקבל את ההפרש בין אופרנד היעד (השני) ואופרנד המקור (הראשון).	sub #3, r1	אויגר r1 מקבל את הפרש בין תוכן האויגר r1 והמספר 3.
lea	lea HELLO, r1 רישי תיבות של HELLO מוכנס לאויגר r1 load effective address. פעלת זה מבצעת טעינה של המعن בזיכרון המצוין על ידי התווית שבאופרנד הראשון (המקור), אל אופרנד היעד, (האופרנד השני).	lea HELLO, r1	המען של תווית HELLO מוכנס לאויגר r1.

#### קבוצת הפקודות השנייה:

ההוראות הדורשות אופרנד אחד בלבד. במקרה זה זוג הסיביות (4-5) חסרות משמעות מכיוון שאין אופרנד מקור (אופרנד ראשון) אלא רק אופרנד יעד (שני). על קבוצה זו נמננות ההוראות הבאות:

not, clr, inc, dec, jmp, bne, red, prm, jsr

פקודת	הסבר פעולה	דוגמא	הסבר דוגמא
not	הפיכת ערכי הסיביות באופרנד (כל סיבית שערכה 0 תהפוך ל-1 והיפך: 1 ל-0). האופרנד יכול להיות אויגר בלבד. אין השפעה על הדגלים.	not r2	r2 ← not r2
clr	אפס את תוכן האופרנד.	clr r2	r2 ← 0
inc	הגדלת תוכן האופרנד באחד.	inc r2	r2 ← r2 + 1

C $\leftarrow$ C - 1	dec C	הקטנת תוכן האופרנד אחד.	dec
PC $\leftarrow$ LINE	jmp LINE	קפיצה בלתי מותנית אל המע מיוצג על ידי האופרנד.	jmp
אם ערך דגל Z באוגר הסטטוס (PSW) הינו 0 או : PC $\leftarrow$ LINE	bne LINE	bne הינו ראשית תיבות של : branch if not equal (to zero). זהי פקודה הستטוטה מוחנית. הערך במצב התוכנית (PC) קיבל את ערך אופרנד היעד אם ערכו של דגל האפס (דגל (Z) באוגר הסטטוס (PSW) הינו 0.	bne
קוד ה-ascii של התו, הנראה מלה מקשיים, יוכנס לאוגר r1.	red r1	קריאה שלתו מותך לוח המקשיים אל האופרנד.	red
התו אשר קוד ה-ascii שלו נמצא באוגר r1 יודפס לקובץ הקלט הסטנדרטי.	prn r1	הדפסת התו שערך ה-ascii שלו נמצא באופרנד, אל קובץ הפלט הסטנדרטי (stdout).	prn
stack[SP] $\leftarrow$ PC SP $\leftarrow$ SP - 1 PC $\leftarrow$ FUNC	jsr FUNC	קריאה לשגרה (סברוטינה). הכנות מצביע התוכנית (PC) לתוכה מחסנית של זמן ריצה והעברות ערך האופרנד למצביע התוכנית (PC).	jsr

### קבוצת הפקודות השלישייה:

מכילה את ההוראות ללא אופרנדים – כלומר ההוראות המורכבות ממיליה אחת בלבד בלבד.

ההוראות השויות לקבוצה זו הן: rts, stop

פקודה	הסבר פעולה	דוגמה	הסבר דוגמא
rts	הוראת חזרה משיגרה. ביצוע הוראות <code>pop</code> על המחסנית של זמן ריצה, והעברת הערך שהוא שם לאוגר התוכנית (PC). הוראה זו מורכבת ממיליה אחת בלבד (בת 12 סיביות). במללה זו החלק המשמעותי הן הסיבות 9-6 המהוות את קוד ההוראה. לשאר הסיבות אין כל חשיבות.	rts	$SP \leftarrow SP + 1$ $PC \leftarrow stack[SP]$
stop	הוראה לעצרת התוכנית. ההוראה מורכבת ממיליה אחת בלבד (בת 12 סיביות). במללה זו החלק המשמעותי הן הסיבות 9-6 המהוות את קוד ההוראה. לשאר הסיבות אין כל חשיבות.	stop	עצירת התוכנית.

### מספר נקודות נוספות לגבי תיאור שפת האסטטלי:

שפה האסטטלי מורכבת ממשפטים (statements) כאשר התו, המפריד בין משפט למשפט, הינו תו 'ט' (טו שורה חדשה). ככלומר, כאשר מסתכלים על הקובץ, רואים אותו כמורכב משורות של משפטיים, כאשר כל משפט מופיע בשורה מסוימת.

ישנם ארבעה סוגי משפטיים בשפה האסטטלי, והם:

סוג המשפט	הסבר כללי
משפט ריק	זהו שורה המכילה בתוכה אך ורק תווים לבנים (white spaces) ככלומר מרכיבת מצירוף של 'ז' ו-' (סימני tab ורווח).
משפט העלה	זהו משפט אשר התו בעמודה הראשונה בשורה בה הוא מופיע הינו תו 'ז' (נקודת פסיק). על האסטטלי להתעלם מהלטין משורה זו.
משפט הנטיה	זהו משפט המנחה את האסטטלי בעת הביצוע. יש מספר סוגים משפטי הנטיה. משפט הנטיה אינו מייצר קוד.
משפט פעולה	זהו משפט המייצר קוד. הוא מכיל בתוכו פעולה שעלה ה-CPU לבצע, וティיר האופרנדים המשתתפים בפעולה.

cut נפרט לגבי סוגי המשפטים השונים.

### משפט הנטיה:

משפט הנטיה הוא בעל המבנה הבא :

בתחליתו יכולה להופיע תוית (label) (התוית חייבת להיות בעלי תחביר חוקי. התחביר של תוית חוקית יתואר בהמשך). לאחר מכן מופיע התו ' .' (נקודה) ובצמוד אליה שם הנטיה. לאחר שם הנטיה היא אופציונלית. לאחר מכן מופיעם מספר פרמטרים שלו (מספר הפרמטרים קבוע בהתאם לסוג הנטיה).

ישנם ארבעה סוגי משפטי הנטיה ו הם :

‘.data’ .1

הפרמטרים של .data. הינו רשימת מספרים חוקיים המופרדים על ידי התו ' .' (פסיק). למשל:

.data   +7 , 17 , 9

יש לשים לב שהפסיקים אינם חייבים להיות צמודים למספרים. בין מספר לפסק ו בין פסיק למספר יכול להופיע מספר כלשהו של רווחים לבנים, או ללא רווחים לבנים כלל. אולם, הפסק חייב להופיע בין הערכים.

משפט הנטיה : ‘.data’ מדריכה את האסמלר להקצת מקום בהמשך חלק תומנת הנתונים (data image) שלו, אשר בו יאוחסנו הערכים המתאים, ולאחר מכן תומנת הנתונים, בהתאם למספר הערכים בראשימה. אם להוראת .data. הייתה תוית זו מקבלת אותה ערך מונה בתמונת הנתונים (לפני הקידום) ומוכנסת אל טבלת הסמלים. דבר זה מאפשר להתייחס אל מקום מסוים בתמונת הנתונים, דרך שם הנטיה.

יש לשים לב: **משפט הנטיה לא מצורפות זוג סיביות E,R,A ותקידוד מלא את כל 12 הסיביות שיש בתא זיכרון.**

כלומר אם נכתב :

XYZ:   +7, -57, 17, 9

movl   XYZ, r1

אז יוכנס לאוגר r1 הערך +7.

לעומת זאת :

lea1 XYZ, r1

יכניס לאוגר r1 את המعن בזיכרון (בחולק הנתונים) אשר בו אוחSEN הערך +7.

‘.string’ .2

הפרמטר של הוראת ‘.string’ הינו מחרוזות חוקית אחת. משמעותה דומה להוראת ‘.data’ . תווי ascii המרכיבים את המחרוזת מקודדים לפי ערכי ה-ascii המתאימים ומוכנסים אל תומנת הנתונים, לפי סדרם. בסוף יוכנס ערך אפס, לסמן סיום מחרוזת. ערך מונה הנתונים יוגדל בהתאם לאורך המחרוזת + 1. אם יש תוית באוטה שורה, אז ערכה יצביע אל המקום בזיכרון, שבו מאוחSEN קוד ה-ascii של התו הראשון במחרוזת, באוטה צורה כפי שנעשה הדבר עבור ‘.data’ .

כלומר משפט הינה:

ABC: .string "abcdef"

מקרה "מערך תוויה" באורך של 7 מילימ' החל מהמען המזוהה עם התווית ABC, ומאתחלת "מערך" זה לערכי ה- ascii של התווים f,e,d,c , b,a מחרוזת תוויה.

### 3. '.entry'

להוראת '.entry' פרמטר אחד והוא שם של תווית המוגדרת בקובץ (מקבלת את ערכה שם). מטרת '.entry' היא להציג על התווית זו כל תווית אשר קטעי אסמבלי, הנמצאים בקבצים אחרים, מתאימים אליה.

לדוגמא:

HELLO:	.entry add1 .....	HELLO #1, r1
--------	-------------------------	-----------------

מודיע שקטע (או קטעי) אסמבלי אחר, הנמצא בקובץ אחר, יתיחס לתווית HELLO.  
הערה: תווית בתחילת שורת entry. חסרת משמעות.

### 4. '.extern'

להוראת '.extern' פרמטר אחד בלבד והוא שם של תווית. מטרת ההוראה היא להציג כי התווית מוגדרת בקובץ אחר וכי קטע האסמבלי, בקובץ זה, עשויה בו שימוש. בזמן הקישור (link) תבוצע בהתאם, בין הערך, כפי שהוא מופיע בקובץ שהגדיר את התווית, לבין ההוראות המשמשות בו, בקבצים אחרים. גם בהוראה זו, תווית, המופיעה בתחילת השורה, אינה חסרת משמעות.

לדוגמא, משפט הינה: '.extern' המתאים למשפט הינה: '.entry' בדוגמא הקודמת תהיה:

.extern HELLO

שורת הוראה:

- שורת הוראה מורכבת מ:  
1. תווית אופציונלית.  
2. שם ההוראה עצמה.  
3. או 1,0 או 2 אופרנדים בהתאם להוראה.

אורכה 80 תווים לכל היוטר.

שם ההוראה נכתב באותיות קטנות (lower case) והיא אחת מ- 16 ההוראות שהוזכרו לעיל.

לאחר שם ההוראה, יכול להופיע האופרנד או האופרנדים.

במקרה של שני אופרנדים, שני האופרנדים מופרדים בטו ' ' (פסיק). קודםם, לא חייבת להיות שומם הצמודה של האופרנדים לטו המפריד או להוראה באופן כלשהו. בלעדן מופיעים רווחים או tabs בין האופרנדים לפסיק ובין האופרנדים להוראה, הדבר חוקי.

להוראה בעלת שני אופרנדים המבנה של:

הווראה אופרנד יעד, אופרנד מקור תווית אופציאונלית

לדוגמא:

HELLO: add1 r7, B

לפקודה בעלת אופרנד אחד המבנה הבא:

הווראה אופרנד תווית אופציאונלית

לדוגמא:

HELLO: bne1 XYZ

להוראה ללא אופרנדים המבנה הבא:

הווראה תווית אופציאונלית

לדוגמא:

END: stop1

אם מופיעות תוויות בשורת ההוראה אזי היא תוכנס אל טבלת הסמלים. ערך התווית יקבע על מקום ההוראה בתחום תומנת הקוד שבונה האסטמבר.

תווית:

תווית חוקית מתחילה באות (גדולה או קטנה) ולאחוריה סדרה כלשבי של אותיות וספרות, שאורכה קטן או שווה 30 תווים. התווית מסתניימת על ידי התוו ': ' (נקודותיים). تو זה אינו מהו חלק ממש התווית. זהו רק סימן המייצג את סופה. כמו כן התווית חיבת להתחיל בעומדה הראשונה בשורה. אסור שיופיעו שתי הגדרות שונות לאותה התווית. התווית שללון הן תוויות חוקיות.

hEllo:

x:

He78902:

שם של הווראה או שם חוקי של רגיסטר אינם יכולים לשמש כשם של תווית.

התווית מקבלת את ערכה בהתאם להקשר בו היא מופיעה. התווית בהוראות 'string', 'data' המתאימים, מקבלת ערך מונה הנטוונם (data counter) המתאים, בעוד שתווית המופיע בשורת ההוראה, מקבלת ערך מונה ההוראות (instruction counter) המתאים.

מספר:

מספר חוקי מתחילה בסימן אופציאוני '-' או '+' ולאחריו סדרה כלשבי של ספרות בסיס עשר. הערך של המספר הוא הערך המוצג על ידי מחרוזת הספרות והסימן. כך למשל -5, 76, 123+. הינם מספרים חוקיים. (אין טיפול במספרים ממשיים).

### מחרוזת:

מחרוזת חוקית היא סדרת תוֹי ascii נראים, המוקפים במרקאות כפולות (המרקאות אינן חשובות כחלק מהמחרוזת). דוגמא למחרוזת חוקית: "hello world".

### **אסמבילר שני מעבריים**

כאשר מקבל האסמבילר קוד לתרגום, עליו לבצע שתי משימות עיקריות: הראשונה היא לזוזות ולתרגם את קוד ההוראה, והשנייה היא לקבוע מיעדים לכל המשתנים והנתונים המופיעים בתוכנית. דוגמא: אם האסמבילר קורא את קטע הקוד הבא:

```

MAIN:    mov1 K, LENGTH
          add1 r2,STR
LOOP:     jmp1 END
          prn1 #5
          sub1 #1, r1
          inc1 r0
          mov1 $$,r3
          bne1 LOOP
END:      stop1
STR:      .string "abcdef"
LENGTH:   .data 6,-9,15
K:        .data 2

```

עליו להחליפ את שמות הפעולה mov, add, jmp, prn, sub, inc, bne, stop בקוד הבינארי השקלול להם במחשב שהגדנו.

כמו כן, על האסמבילר להחליפ את הסמלים K,STR, LENGTH, MAIN, LOOP, END במערכות של האטרים שהוקטו לנוטונם, ובמשמעותם של ההוראות המתאימות.

נניח לרגע שקבע הקוד לעלה יאוחSEN (הוראות ונתונם) בקטע זיכרונו החל ממען 0100 (בבסיס 10) בזיכרון. הערה: במקורה זה קיבל את ה"תרגום" הבא:

**لتשומת לב:** המקיפים המופיעים בקידוד הבינרי הם רק לצורך הפרדה של השדות השונים בקידוד וונעדו לשם המתחשה בלבד.

Label	Decimal Address	Base 4 Address	Command	Operands	Binary machine code
MAIN:	0100	1210	mov1	K, LENGTH	10-0000-10-01-00
	0101	1211		כתובת של K	0010000010-10
	0102	1212		כתובת של LENGTH	0001111111-10
LOOP:	0103	1213	add1	r2, STR	10-0010-11-01-00
	0104	1220		קידוד מספר האוצר	00010-00000-00
	0105	1221		כתובת של STR	0001111000-10
END:	0106	1222	jmp1	END	01-1001-00-01-00
	0107	1223		כתובת של END	0001110111-10
PRN:	0108	1230	prn1	#-5	01-1100-00-00-00
	0109	1231		-5	1111111011-00
SUB:	0110	1232	sub1	r1,r4	10-0011-11-11-00
	0111	1233		קידודי מספרי האוצרים	00001-00100-00
INC:	0112	1300	inc1	K	01-0111-00-11-00
	0113	1301		כתובת של K	0010000010-10
MOV:	0114	1302	mov1	\$\$,r3	00-0000-10-11-00

	<b>0115</b>	1303			כתובת של K קידוד מספר האוצר	<b>0010000010-10</b>
	<b>0116</b>	1310				<b>00000-00011-00</b>
	<b>0117</b>	1311	bne1	LOOP		<b>01-1010-00-01-00</b>
	<b>0118</b>	1312			כתובת של LOOP	<b>0001101010-10</b>
<i>END:</i>	<b>0119</b>	1313	stop1			<b>00-1111-00-00-00</b>
<i>STR:</i>	<b>0120</b>	1320	.string	“abcdef”		<b>000001100001</b>
	<b>0121</b>	1321				<b>000001100010</b>
	<b>0122</b>	1322				<b>000001100011</b>
	<b>0123</b>	1323				<b>000001100100</b>
	<b>0124</b>	1330				<b>000001100101</b>
	<b>0125</b>	1331				<b>000001100110</b>
	<b>0126</b>	1332				<b>000000000000</b>
<i>LENGTH:</i>	<b>0127</b>	1333	.data	6,-9,15		<b>000000000110</b>
	<b>0128</b>	2000				<b>111111110111</b>
	<b>0129</b>	2001				<b>0000000001111</b>
<i>K:</i>	<b>0130</b>	2002	.data	2		<b>0000000000010</b>

אם האסםבלר מחזיק בטבלה שבה רשומים כל שמות הפעולה של ההוראות והקודים הבינאריים המתאים להם, איזי שמות הפעולה ניתנים להמרת בקשות. כאשר נקרא שם פעולה, אפשר פושט לעיינו בטבלה ולמצוא את הקוד הבינארי השוכן.

כדי לעשות את אותה פעולה לגבי מענים סטטיטיים, יש צורך לבנות טבלה דומה. אולם הקודים של הפעולות יזועים מראש, ואילו היחס בין הסטטיטים, בשימוש המתכונת, לבין מעני התוצאות שלהם בזיכרנו. אין ידוע, עד אשר התוכנית מköודדת ונקראת על ידי המחשב.

בדוגמא שלפנינו, אין האסטמבלר יכול לדעת שהסמל LOOP משוויך למספר 0106 (עשרוני), אלא רק לאחר שהתכוינית נקבעה כולム.

לכן יש שתי פעולות נפרדות, שצריך לבצע כל הסמלים, שהוגדרו ביד המתכונת. הראשונה היא לבנות טבלה של כל הסמלים והעריכים המספריים המשווים להם, והשנייה היא להחליף את כל הסמלים, המופיעים בתוכנית בשדה המعنן, בערכיהם המספריים. שלבים אלה קשורים בשתי סדריות, מעבירים, של קוד המקור. במעבר הראשון נבנית טבלת סמלים בזיכרון, שמוגדרים בה משאים לכל הסמלים. בדוגמא דילג, טבלת הסמלים לאחר מעבר ראשון היא:

סמל	ערך דצימלי	ערך בבסיס 4
MAIN	100	1210
LOOP	106	1222
END	119	1313
STR	120	1320
LENGTH	127	1333
K	130	2002

במעבר השני נעשית ההחלהפה, כדי לתרגם את הקוד לבינארי. עד אותו זמן צריכים הערכים של כל הסמלים, להיות כבר ידועים.

עליך לשים לב: שני המUberים של האסSEMBLER נועשים עד לפניה שוכנית המשתמש נתעננת לזכרוון, לצורך הביצוע. כלומר, התרגום נעשה בזמן אסSEMBLY, שהוא הזמן שבו נמצאת הבקרה בידי האסMBLER.

לאחר השלמת תהליך התרגומים, יכולת תוכנית המשמש לעבור לשלב הקישור/טעינה ולאחר מכן לשלב הביצוע. הביצוע נעשה בזמן ריצה.

## המעבר הראשוני

במעבר הראשוני נדרשים כללים כדי לקבוע איזה ערך מען ישיק לכל סמל. העיקרונו הבסיסי הוא לספור את המיקומות בזיכרון, שאוטם צורכת כל הוראה כאשר היא נكرة. אם כל הוראה תיטען לאחר האסטטטלי, לאטר העוקב של אטר ההוראה הקודמת, תציג ספריה כזאת את מען ההוראה. הספריה נעשית על ידי האסטטטלי ומוחזקת באוגר הנקרה מונה אטרים (IC). ערכו התחלתי הוא 0, וכן נתען משפט ההוראה הראשונית במען 0. הוא משתנה על ידי כל הוראה, או הוראה מודומה, המקרה מקום. לאחר שהאסטטטלי קובע מהו אורך ההוראה, תוכנו של מונה האטרים עולה במספר הבתים, הנתפסים על ידי ההוראה, וכך הוא מביע על התא הריך הבא.

כאמור, כדי לקודד את ההוראות בשפה מכונה, מוחזק האסטטטלי טבלה, שיש בה קוד מתאים לכל ההוראה. בזמן התרגומים מוחלף האסטטטלי בכל ההוראה בקוד שלה. אך פולת ההחלפה אינה כה פשוטה. יש הרבה ההוראות המשמשות בצורות מיון שונות. אותה ההוראה יכולה לקבל משמעותות רבות, בכל אחת מצורות המיון, וכך יתאים לה קודים שונים. לדוגמה, הוראות ההזזה אסמו יכולות להתייחס להעברת תוכן תא זיכרון לאוגר, או להעברת תוכן אוגר לאור, וכן הלאה. ככל צורה זו את של אסמו מתאים קוד שונה.

על האסטטטלי לסרוק את שורת ההוראה בשלמותה, ולהחליט לגבי קוד ההוראה לפי האופרנדים שלה. בדרך כלל מתחלק קוד ההוראה המתורגם לשדה קוד ההוראה וסדרות נוספים, המכילים מידע לגבי שיטות המיון.

במחשב שלנו קיימת גמישות לגבי שיטות המיעון של שני האופרנדים. הערה: דבר זה לא מחייב לגבי כל מחשב. ישנו מחשבים בהם כל הפקודות הן בעלות אופרנד יחיד (והפעולות מתבצעות על אופרנד זה ואוגר קבוע) או מחשבים המאפשרים מבחוץ של שיטות מייען עבור אופרנד אחד והואופרנד השני חייב להיות אוגר בלבד, או מחשבים בעלי 3 אופרנדים, כאשר האופרנד השלישי משמש לאחסון תוצאה הפעולה.

כאשר נתקל האסטמבלר בתווית המופיעה בתחילת השורה, הוא יודע שלפני הגדרה של תווית, ואז ניתן לה מען – תוכנו הנוכחי של מונה התארירים. כך מקבלות כל התוויות את מעניהם בעת ההגדירה. תוויות אלה מוכנסות לטבלת הסמלים, המכילה בנוסף לשם התווית גם את מענה ומאפיינים נוספים שללה, כגון סוג התווית. כאשר תהיה התייחסות לתווית זאת בשדה המعن של הוראה, יוכל האסטמבלר לשולף את המען המתאים מהטבלה.

בידוע, מתכנת יכול להתייחס גם לטלם, ללא הוגדר עד כה בתוכנית, אלא רק לאחר מכן. לדוגמה:  
פקודת הסתעפות למען, המופיע בהמשך הקוד:

bne1 A

.

A: .....

כאשר מגיע האסטמבלר לשורה זו (bne1), הוא עדיין לא הקצה מען לתווית A וכן אינו יכול להחליף את הסמל A בمعנו בזיכרונו. נראה עתה כיצד נפתרת בעיה זו.

#### מעבר שני

בעת המעבר הראשון, אין האסטמבלר יכול להשלים בטבלה את מעני הסמלים שלא הוגדרו עדין, והוא מצין אותם באפסים. רק לאחר שהאסטמבלר עבר על כל התוכנית, כך שכל התוויות הולנסו כבר לטבלת הסמלים, יכול האסטמבלר לחילוף את התוויות, המופיעות בשדה המعن של הוראה, בمعنىهن המתאימים. לשם כך עובר האסטמבלר שנית על כל התוכנית, ומחילוף את התוויות, המופיעות בשדה המعن, בمعنىهن המתאימים מתוך הטבלה. זהו המעבר השני, ובסיומו תהיה התוכנית מתורגמת בשלמותה.

#### אסטמבלר של מעבר אחד

יש תוכניות אסטמבלר שאינן מבצעות את המעבר השני, והחלפת המענים נעשית בהם בדרך אחרת: בזמן המעבר הראשון שומר האסטמבלר טבלה שבה נשמר עבור כל תווית, מען ההוראה שיש בה התייחסות אל התווית בחלק המعن.

דוגמא:

נניח שבמען 400 בתוכנית מוגדרת התווית TAB.  
נניח גם שבמען 300 מופיע r1 add TAB,  
ובمعنى 500 מופיע TAB jmp.

```
300:      add1 TAB, r1
.....
400:      TAB: .....
.....
500:      jmp1 TAB
```

האסטמבלר יקצת כניסה לטבלה לתווית TAB, ויניח בה את המענים 301 ו-501. (המענים הנשמרים הם 301 ו-501 ולא 300 ו-500 מכיוון שירות ההוראה מופיע בכתובות אלה, והמילה

הנוספת מופיעה בכתובות שבהן מיד לאחר מכן. הערזה: המענים יכולים להישמר גם בנסיבות נפרדות, הדבר תלויה בנסיבות היישום. בסוף המעבר הראשון, מלא האסטבלר את המענים החסרים בתרגום הקוד מתוך הטבלה. היתרונו בשיטה זו הוא כפונה, שהאסטבלר חוסך את המעבר השני על כל התוכנית.

#### **הפרצת הוראות וגთוניות**

לכמה תוכניות אסטבלר יש מוני אטרים אחדים. אחד השימושים לכך הוא הפרדת הקוד והנתונים לקטעים שונים בזיכרון, שיטה שהיא עדיפה על פני הצמדה של הגדרות הנתונים להוראות המשמשות בהן.

אחת הסכנות הטמונה באין הפרדת הקוד מהנתונים היא, שלפעמים עלול המעבד, בעקבות שגיאה קלה, לנסתות לבצע את הנתונים. שגיאה שכולה לגרום זאת היא, למשל, השמטת הוראה עצירה או הסתעפות לא נכונה. אם הנתון שאותו מסה המעבד לבצע אכן מהוות קוד של הוראה חוקית, תתקבל מיד הודעה שגיאה. אך אילו הנתון נראה כהוראה חוקית, הינה הטעיה חמורה יותר, משום שהשגיאה לא הייתה מתגלית מיד.

**בתוכנית האסטבלר, עליליך למש, יש להפריד בין קטע הנתונים לקטע ההוראות, ככלומר בקביצי הפלט תהיה הפרדה של קוד וגთוניות, ואילו בקובץ הקלט שניינו לתוכנית אין חובה שתמיהה הפרדה.**

#### **גילוי שגיאות אסטבלר**

האסטבלר יכול לגלו שגיאות בתחריר של השפה, כגון הוראה שאינה קיימת, מספר אופרנדים שגוי, אופרנד שאינו מתאים להוראה ועוד. כן מודיא האסטבלר לכל הסמלים מוגדרים עם אחת בדיק. מכאן שאת השגיאות, המתגלוות בידי האסטבלר, אפשר לשיק בדרך כלל לשורת קלט מסוימת. אם, לדוגמה, ניתנו שני מענים בהוראה שאמור להיות בה רק מען יחיד, האסטבלר עשו ליתג הודעת שגיאה בנוסח "ייתר מדי מענים". בדרך כלל מודפסת הודעה כזאת בתדף הפלט, באורה שורה או בשורה הבאה, אם כי יש תוכניות אסטבלר המשמשות בסימנון מקוצר כלשהו, ומפרטות את השגיאות בסוף התוכנית.

הטבלה הבאה מכילה מידע על שיטות מייען חוקיות, עבר או פרנד המקור, ואופרנד היעד, עבר הפקודות השונות הקיימות בשפה הנתונה:

פעולה	שיטות מייען חוקיות עבר	שיטות מייען חוקיות מקור או פרנד יעד
mov	, 0,1,2,3,	1,3
cmp	, 0,1,2,3,	, 0,1,2,3,
add	, 0,1,2,3,	, 1,3,
sub	, 0,1,2,3,	, 1,3,
not	אין אופרנד מקור	, 1,3,
clr	אין אופרנד מקור	1, 3,
lea	1	, 1,3,
inc	אין אופרנד מקור	, 1,3,
dec	אין אופרנד מקור	, 1,3,
jmp	אין אופרנד מקור	1,2,3,
bne	אין אופרנד מקור	1,2,3,
red	אין אופרנד מקור	, 1,2,3,
prn	אין אופרנד מקור	, 0,1,2,3,
jsr	אין אופרנד מקור	, 1
rts	אין אופרנד מקור	אין אופרנד יעד
stop	אין אופרנד מקור	אין אופרנד יעד

שגיאות נוספות אפשריות הן פקודה לא חוקית, שם רגיסטר לא חוקי, תווית לא חוקית וכו'.

## אלגוריתם כללי

להלן נציג אלגוריתם כללי למעבר הראשוני ולמעבר השני: אנו נניח כי הקוד מחולק לשני אזורים, אזור ההוראות (code) ואזור הנתונים (data). נניח כי לכל אזור יש מונה משלה, ונסמן באותיות IC (מונה ההוראות - Instruction counter) ו-DC (מונה הנתונים - Data counter). האות L מסמן את מספר המילים שתופסת הוראה.

### מעבר ראשון

- .1.  $DC \leq 0, IC \leq 0$ .
- .2. קרא שורה.
- .3. האם השזהה הראשון הוא סמל? אם לא, עברו ל-5.
- .4. הצב דגל "יש סמל".
- .5. האם זהה הוראה מדומה (הנחיה לאחסון נתונים, כולם, האם הנחית data או ?.string)? אם לא, עברו ל-8.
- .6. אם יש סמל, הכנס אותו לטבלת הסמלים עם סימון (סמל data). ערכו יהיה DC.
- .7. זהה את סוג הנתונים, אחסן אותם בזיכרון, עדכן את מונה הנתונים בהתאם לארכם, חזרו ל-2.
- .8. האם זו הנחית entry או הנחית ?.entry? אם לא, עברו ל-11.
- .9. האם זהה הצהרת extern? אם כן, הכנס את הסמלים לטבלת הסמלים החיצוניים, ללא מען.
- .10. חזרו ל-2.
- .11. אם יש סמל, הכנס אותו לטבלת הסמלים עם סימון (סמל code). ערכו יהיה IC.
- .12. חפש בטבלת ההוראות, אם לא מצאת – הודיע על שגיאה בקוד הוראה.
- .13.  $IC \leq L + IC$ .
- .14. חזרו ל-2.

### מעבר שני

- .1.  $IC \leq 0$ .
- .2. קרא שורה. אם סיימת, עברו ל-11.
- .3. אם השזהה הראשון הוא סמל, דרג עלייו.
- .4. האם זהה הוראה מדומה (data, string, ?.extern, ?.entry)? אם כן, חזרו ל-2.
- .5. האם זהה הנחיה entry? אם לא, עברו ל-7.
- .6. זהה את הנחיה. השלים את הפעולה המתאימה לה. אם זאת הנחית entry. סמן את הסמלים המתאים כ-entry. חזרו ל-2.
- .7. הערך את האופרנדים, חפש בטבלת ההוראות, החל את הוראה בקוד המתאים.
- .8. אחסן את האופרנדים החיל מהבית הבא. אם זהו סמל, מצא את המען בטבלת הסמלים, חשב מענים, קודד שיטת מעון.
- .9.  $IC \leq IC + L$ .
- .10. חזרו ל-2.
- .11. שמור על קבוע נפרד את אורך התוכנית, אורך הנתונים, טבלת סמלים חיצוניים, טבלת סמלים עם סימוני נקודות כניסה.

נפעיל אלגוריתם זה על תוכנית הדוגמא שראינו קודם :

```
MAIN:    mov1 K, LENGTH
          add1 r2,STR
LOOP:     jmp1 END
          prn1 #5
          sub1 #1, r1
          inc1 K
          mov1 $$,r3
          bne1 LOOP
END:      stop1
STR:      .string "abcdef"
LENGTH:   .data 6,-9,15
K:        .data 2
```

בוצע עתה מעבר ראשוני על הקוד הנוכחי. בוצע במעבר זה גם את החלפת ההוראה בקוד שלו. כמו כן נבנה את טבלת הסמלים. את החלקיים שעדיין לא מתורגםים בשלב זה, נשאיר כמותותיהם. נניח שהקוד ייטע החל מהמען 100 (בבסיס 10).

<i>Label</i>	Decimal Address	Base 4 Address	Command	Operands	Binary machine code
<i>MAIN:</i>	0100	1210	<b>mov1</b>	K, LENGTH כתובת של K כתובת של LENGTH	<b>10-0000-01-01-00</b> ????????????? ?????????????
	0101	1211			?????????????
	0102	1212			?????????????
<i>LOOP:</i>	0103	1213	<b>add1</b>	r2, STR כתובת של r2, STR קיזוד מספר האוגר	<b>10-0010-11-01-00</b> <b>00010-00000-00</b> ?????????????
	0104	1220			
	0105	1221			
<i>END:</i>	0106	1222	<b>jmp1</b>	END כתובת של END	<b>01-1001-00-01-00</b> ?????????????
	0107	1223			
	0108	1230	<b>prn1</b>	#-5 המספר -5	<b>01-1100-00-00-00</b> <b>1111111011-00</b>
<i>STR:</i>	0109	1231			
	0110	1232	<b>sub1</b>	r1,r4 כתובת של r1,r4 קיזודי מספרי האוגרים	<b>10-0011-11-11-00</b> <b>00001-00100-00</b>
	0111	1233			
<i>END:</i>	0112	1300	<b>inc1</b>	K כתובת של K	<b>01-0111-00-01-00</b> ?????????????
	0113	1301			
	0114	1302	<b>mov1</b>	\$\$,r3 כתובת של \$\$,r3 קיזוד מספר האוגר	<b>00-0000-01-11-00</b> ????????????? <b>00000-00011-00</b>
<i>END:</i>	0115	1303			
	0116	1310			
	0117	1311	<b>bne1</b>	LOOP כתובת של LOOP	<b>01-1010-00-01-00</b> ?????????????
<i>END:</i>	0118	1312			
	0119	1313	<b>stop1</b>		<b>00-1111-00-00-00</b>
	0120	1320	<b>.string</b>	“abcdef”	<b>000001100001</b>
<i>LENGTH:</i>	0121	1321			<b>000001100010</b>
	0122	1322			<b>000001100011</b>
	0123	1323			<b>000001100100</b>
	0124	1330			<b>000001100101</b>
	0125	1331			<b>000001100110</b>
	0126	1332			<b>000000000000</b>
<i>LENGTH:</i>	0127	1333	<b>.data</b>	6,-9,15	<b>0000000000110</b> <b>11111110111</b> <b>000000001111</b>
	0128	2000			
	0129	2001			
<i>K:</i>	0130	2002	<b>.data</b>	2	<b>000000000010</b>

טבלת הסמלים :

סמל	ערך דצימלי	ערך בבסיס 4
MAIN	100	1210
LOOP	106	1222
END	119	1313
STR	120	1320
LENGTH	127	1333
K	130	2002

מבצע עתגה את המעבר השני ונרשום את הקוד בצורתו הסופית:

Label	Decimal Address	Base 4 Address	Command	Operands	Binary machine code
MAIN:	0100	1210	mov1	K, LENGTH כתובת של K	10-0000-01-01-00 0010000010-10 0001111111-10
	0101	1211		כתובת של LENGTH	
	0102	1212			
	0103	1213	add1	r2, STR קידוד מספר האגר	10-0010-11-01-00 00010-00000-00 כתובת של STR
	0104	1220			
	0105	1221			
LOOP:	0106	1222	jmp1	END כתובת של END	01-1001-00-01-00 0001110111-10
	0107	1223			
	0108	1230	prn1	#-5 המספר -5	01-1100-00-00-00 1111111011-00
	0109	1231			
	0110	1232	sub1	r1,r4 קידודי מספרי האוגרים	10-0011-11-11-00 00001-00100-00
	0111	1233			
	0112	1300	inc1	K כתובת של K	01-0111-00-01-00 0010000010-10
	0113	1301			
	0114	1302	mov1	\$\$,r3 כתובת של K	00-0000-01-11-00 0010000010-10
	0115	1303			
	0116	1310			
	0117	1311	bne1	LOOP כתובת של LOOP	01-1010-00-01-00 0001101010-10
	0118	1312			
	END:	0119	stop1		00-1111-00-00-00
	STR:	0120	.string	“abcdef”	000001100001
	0121	1321			000001100010
	0122	1322			000001100011
	0123	1323			000001100100
	0124	1330			000001100101
	0125	1331			000001100110
	0126	1332			00000000000000
LENGTH:	0127	1333	.data	6,-9,15	0000000000110 11111110111 000000001111
	0128	2000			
	0129	2001			
K:	0130	2002	.data	2	000000000010

לאחר סיום עבודות תוכנית האסמבילר, התוכנית נשלחת אל תוכנית הקישור/טעינה.

תפקידיה של תוכנית זו הן:

1. להקצות מקומות בזיכרון עבור התוכנית (allocation).
2. לגרום לקישור נכון בין הקבצים השונים של התוכנית (linking).
3. לשנות את כל המענינים בהתאם למקומות הטעינה (relocation).
4. להטיען את הקוד פיסית לזכרון (loading).

לאណון כאן בהרחבה באופן עבודות תוכנית הקישור/טעינה ( כאמור, אינה למימוש בפרויקט זה)

לאחר עבודות תוכנית זו, התוכנית טעונה בזכרון ומוכנה לריצה.

כעת נעיר מספר העזרות ספציפיות לגבי המימוש שלכם:

על תוכנית האסמבילר שלכם לקבל כארגומנטים של שורת פקודה (command line arguments) רשיימה של קבוע טקסט, בהם רשומות הוראות בתחביר של שפת האסמבילר, שהוגדרה לעלה. עבור כל קבוע יוצר האסמבילר קבוע מטרה (object). כמו כן ייווצר (עבור אותו קבוע) קבוע

, באם המקור (source) הציגו על משתנים חיצוניים, וקובץ entries , באם המקור (source) הציגו על משתנים מסוימים בעל נקודות כניסה.

קובץ המקור של האסsembler חייבים להיות בעלי הסיומת "as". השמות x.as , y.as , hello.as הם שמות חוקיים. הפעלת האסsembler על הקבצים הללו נעשית ללא ציון הסיומת. לדוגמה: אם תוכנית האסsembler שלנו נקראת assembler , אזי שורת הפקודה הבאה:

```
assembler x y hello
```

תגרום לכך שתוכנית האסsembler שלנו תקרא את הקבצים :

האסsembler יוצר את קבצי-object , קבצי entries וקבצי-externals על ידי ליקוחם שם הקובץ כפי שהוא בשורת ההוראה והוספת הסיומת "ob" עבור קובץ ה-object , סיומת "ent" עבור קובץ ה-entries , וסיומת "ext" עבור קובץ ה-externals .

#### מבנה כל קובץ יתואר בהמשך.

לדוגמא: הפקודה : assembler x entries/ext.x.xent . ואות הקבצים x.ext ו-x.xent הם קיימיםentries/externals בקובץ .

העבודה על קובץ מסוים נעשית בצורה הבאה:

האסsembler מחזיק שני מערכיים, שייקראו להן מערך הקוד ומערך הנתונים. מערכים אלו נתונים למשהה תמונה של זיכרון המכונה (גודל כל כניסה בזיכרון זהה לגודלה של מילת מכונה – 12 סיביות). במערך הקוד מכיסס האסsembler את הקידוד של הוראות המכונה בהן הוא נתקל במהלך האסsembler. במערך הנתונים מכיסס האסsembler נתונים המתקבלים תוך כדי האסsembler על ידי data.string .

לאסsembler יש שני מונחים: מונה ההוראות (IC) ומונה הנתונים (DC). מונחים אלו מצביעים על המקום הבא הפניו במערכות לעיל בהתאם. כמשמעותו האסsembler את פעולתו על קובץ מסוים שני מונחים אלו מואופסים. במקרה יש לאסsembler בטבת סמלים, אשר בה נשמרות המשתנים, בהם נתקל האסsembler במהלך ריצתו על הקובץ. לכל משתנה נשמרים שמו, ערכו וטיפוסו ( external relocatable ) .

#### **אוף פועלות האסsembler**

האסsembler קורא את קובץ המקור שורה אחר שורה, מחליט מהו טיפוס השורה (הערה, פעולה, הנחיה או שורה ריקה) ופועל בהתאם.

.1. שורה ריקה או שורת הערה: האסsembler מתעלם מן השורה וועבר לשורה הבאה.

.2. שורת פעולה:

כאשר האסsembler נתקל בשורת פעולה זהה מחייבת מהי הפעולה, מהי שיטות המיון ומי הם האופרנדים. (מספר האופרנדים אותו הוא מוחפש נקבע בהתאם לפעולה אותה הוא מצא).

האסsembler קובע לכל אופרנד את ערכו בצורה הבאה:

• אם זה אוגר – ערכו הוא מספר האוגר.

• אם זו תווית – ערכו הוא הערך שלה כפי שהוא מופיע בטבת הסמלים.

• אם זה מספר (מיון ישיר) – ערכו הוא הערך של המספר.

קביעת שיטת המיעון נעשית בהתאם לתחביר של האופרנד, כפי שהוא מתואר בחלק העוסק בשיטות המיעון. למשל, התו '# מצין מיעון מיידי, תווית מצינית מיעון ישיר, שם של אוגר מצין מיעון אוגר.

משמעותם של אוגר ומיידי: ערך שדה האופרנד הינו ערך תווית המשתנה, כפי שהוא מופיע בטבלת הסמלים.

לאחר שהאסמבלר החליט לגבי הניל (פעולה), שיטת מיעון אופרנד מקור, שיטות מיעון אופרנד יעד, אוגר אופרנד מקור, אוגר אופרנד יעד, האם נדרשת מילה נוספת עבור אופרנד מקור באם יש, האם נדרשת מילה נוספת עבור אופרנד יעד באם יש) הוא פועל באופן הבא:

אם זהה הוראה בעלת שני אופרנדים, אזי האסמבller מכניס למערך הקוד, במקומות אליו מציבע מונה ההוראות, שיטות הייצוג של הוראות המכונה כפי שתוארו קודם לכן) את קוד הפעולה, שיטות המיעון, ואת המידיע על האוגרים. בנוסף הוא "משרין" מקום עבור מספר המילים הנוספות, הנדרשות עבור פקודת זו ומגדיל את מונה הקוד בהתאם.

אם ההוראה היא בעלת אופרנד אחד בלבד, כלומר האופרנד הראשון (אופרנד המקור) אינו מופיע, איזה התרגומים הינו זהה לחוטין, כמעט העבודה שסיבות מיעון המקור במללה הראשונה המכונסת לאי-זיכרון (האמורות לייצג את המידע על אופרנד המקור) יכולות להיות בעלות כל ערך אפשרי מכיוון שערך זה אינו משמש כלל את ה-CPU.

אם ההוראה היא ללא אופרנדים (stop, rts), איזי למקום במערך הקוד, שלו מציבע מונה ההוראות, יוכנס מספר אשר מקודד אך ורק את קוד ההוראה של הפעולה. שיטות המיעון ומידע על האוגרים של אופרנדי המקור והיעד, יכולים להיות בעלי ערך כלשהו ללא הגבלה.

אם לשורת הקוד קיימת תווית, איזי התווית מכונסת אל בטבלת הסמלים תחת השם המתאים, ערכיה הוא ערך מונה ההוראות לפני קידוד ההוראה. טיפוסה הוא relocatable.

3: שורות הוראה:  
כאשר האסמבller נתקל בהנחיה הוא פועל בהתאם לסוג שלה, באופן הבא:

I. '.data'.  
האסמבller קורא את רשימת המספרים, המופיעות לאחר '.data', הוא מכניס כל מספר שנקרא אל מערך הנתונים ומקודם את מציבע הנתונים באחד, עבור כל מספר שהוכנס.  
אם ל-'data', יש תווית לפניה, איזי תווית זו מכונסת לטבלת הסמלים. היא מקבלת את הערך של מונה הנתונים, לפני שהנתונים הוכנסו אל תוך הקוד + אורך הקוד הכללי. הטיפוס של הוא relocatable, והגדרתה ניתנה בחלק הנתונים.

II. '.string'.  
התנהגות לגבי '.string'. דומה לו של '.data', אלא שקודם-h-ascii של התווים הנקרים הם אלו המכונסים אל מערך הנתונים. לאחר מכן מוכנס הערך 0 (אפס, המציין סוף מחזורת) אל מערך הנתונים. מונה הנתונים מוקודם באורך המחרוזת + 1 (כי גם האפס תופס מקום). ההנחה לגבי תווית המופיעעה בשורה, זהה להנחהות במקרה של '.data'.

III. '.entry'.  
זהי בקשה מן האסמבller להכניס את התווית המופיעעה לאחר '.entry', אל קובץ ה-entries. האסמבller רושם את הבקשה ובסיום העבודה, התווית הניל תירשם בקובץ ה-entries. באה להזכיר על תווית שנעשה בה שימוש בקובץ אחר, וכי על תוכנית הקישור להשתמש במידע המצויב בקובץ ה-entries ובקובץ ה-externals כדי להתאים בין ההתייחסויות ל-externals.

IV. '.extern'.  
זהי בקשה הבאה להציג על משתנה המוגדר בקובץ אחר, אשר קטע האסמבלי בקובץ עכשווי יעשה בו שימוש. האסמבller מכניס את המשנה המבוקש אל בטבלת הסמלים. ערכו הוא אפס (או כל ערך אחר), טיפוסו הוא external, היקן ניתנה הגדרתו אין יודעים (וain זה משנה עבור האסמבller).

יש לשים לב! בפועל או בהנחה אפשר להשתמש בשם של משתנה, אשר הaczירה עליו ניתנת בהמשך הקובץ (אם באופן עקיף על ידי תווית ואם באופן מפורש על ידיtern.(extem).

### פורמט קובץ ה-object

**האSEMBLER בונה את תומנות זיכרונו המבוגה כך שקידוד החראה הראשונה מקובץ האSEMBLER יכנן**  
**למען 100 (בבסיס 10)** בזיכרונו, קידוד החראה השנייה למען לאחר החראה הראשונה (מען 101 או 102 או 103, תלוי באורך החראה הראשונה) וכך להלאה עד לתרגום החראה האחורונה. מיד לאחר קידוד החראה האחורונה, מכילה תומנות הזיכרונו את הנתונים שנבנו על ידי החראות 'data.' ו-'string.'. הנתונים שייחו ראשונים הם הנתונים המופיעים ראשונים בקובץ האSEMBLER, וכך הלאה. התניות הזיכרונו שבונה האSEMBLER. עקרונית פורמט של קובץ object הינו תומנת הזיכרונו הניל.

קובץ object מורכב משורות שורות של טקסט, השורה הראשונה מכילה (בסיס 4) את אורך הקוד (במילוט זיכרונו) ואת אורך הנתונים (במילוט זיכרונו). שני המספרים מופרדים ביניהם על ידי רווח. השורות הבאות מתארות את תוכן הזיכרונו (שוב, בסיס 4)

במשך מופיע קובץ object לדוגמא שלו: j.ek.ks.the המתאים ל-ps.as

בנוסח עבור כל תא זיכרונו המכיל הוראה (לא data) מופיע מידע עבור תכנית הקישור. מידע זה ניתן ע"י 2 הסיביות הימניות של הקידוד (שדה ה-E,R,A)

האות 'A' מצינית את העובדה שתוכן התא הינו אבסולוטי (absolute) ואינו תלוי היקן באמת יטען הקובץ (האSEMBLER יוציא מתוך הנחה שהקובץ נתען החל מען אפס). במקרה כזו 2 הסיביות יכילו את הערך 00

האות 'R' מצינית שתוכן תא הזיכרונו הינו relocatable ויש להוסיף לתוכן התא את ההיסט (Offset) המתאים (בהתאם למקום בו יטען הקובץ באופן מעשי). ה-offset הינו מען הזיכרונו שבו תטען, למעשה, החראה, אשר האSEMBLER אומר שעלייה להטען בהיעטן במען אפס. במקרה כזו 2 הסיביות יכילו את הערך 10

האות 'E' מצינית שתוכן תא הזיכרונו הינו external וכי תכנית הקישור תזאג להכנסת הערך המתאים. במקרה כזו 2 הסיביות יכילו את הערך 01

### קובץ ה-entries

קובץ ה-entries בני משורות טקסט. כל שורה מכילה את שם ה-entry וערכה, כפי שהושב עבור אותו קובץ (ראה לדוגמה את הקובץ ent.ps.ek.ks המתאים לקובץ האSEMBLER ששמו ps.as).

### קובץ externals

קובץ ה-externals בניי אף הוא משורות טקסט. כל שורה מכילה את שם ה-external ואת המען בזיכרונו, שבו יש התיחסות למשתנה חיצוני זה (לדוגמה הקובץ ps.ext.ks.ek.ks המתאים לקובץ האSEMBLER ששמו ps.as).

להלן קובץ PS.AS לדוגמא :

```
; file ps.as

.entry LOOP
.entry LENGTH
.extern L3
.extern W
MAIN:    mov2 K, W
          add1 r2,STR
          mov1 $$,r4
LOOP:     jmp1 L3
          prn1 #-5
          sub1 $$, r1
          inc1 r0
          mov2 $$,r3
          bne1 L3
END:      stop1
STR:      .string "abcdef"
LENGTH:   .data 6,-9,15
K:        .data 2
```

הקובץ שלහן הוא קובץ object אשר שמו בעל סיומת '.ob'. זהו קובץ שהתקבל מהפעלת התוכנית על קובץ האסמבילר דלעיל. להלן דוגמת הקידוד לביטים ולאחריה פורמט קובץ OB.  
כל תוכן הקובץ מיוצג על ידי מספרים בסיסי <sup>4</sup>.

:ps.ob :קובץ

<i>Label</i>	<i>Decimal Address</i>	<i>Base 4 Address</i>	<i>Command</i>	<i>Operands</i>	<i>Binary machine code</i>
MAIN:	0100	1210	mov2	K, W	10-0000-01-01-00
	0101	1211		כתובת של K	0010001001-10
	0102	1212		כתובת של W	0000000000-01
	0103	1213		קיזוד הפקודה פעם נספה	10-0000-01-01-00
	0104	1220		כתובת של K	0010001001-10
	0105	1221		כתובת של W	0000000000-01
	0106	1222	add1	r2, STR	10-0010-11-01-00
	0107	1223		קידוד מספר האיגר	00010-00000-00
	0108	1230		STR	0001111111-10
	0109	1231	mov1	S\$,r4	10-0000-11-11-00
	0110	1232		r2 - r4	00010-00100-00
LOOP:	0111	1233	jmp1	L3	01-1001-00-01-00
	0112	1300		כתובת של L3	0000000000-01
	0113	1301	prn1	#-5	01-1100-00-00-00
	0114	1302		-5	1111111011-00
	0115	1303	sub1	\$\$,r4	10-0011-00-11-00
	0116	1310		-5	1111111011-00
	0117	1311		קידוד האיגר	00000-00100-00
	0118	1312	inc1	r0	01-0111-00-11-00

	<b>0119</b>	1313		קידוד האוגר	00000-00000-00
	<b>0120</b>	1320	<b>mov2</b>	\$\$,r3	<b>00-0000-11-11-00</b>
	<b>0121</b>	1321		קידוד האוגרים	00000-00011-00
	<b>0122</b>	1322		קידוד הפקודה פעם נוספת	<b>00-0000-11-11-00</b>
	<b>0123</b>	1323		קידוד האוגרים	00000-00011-00
	<b>0124</b>	1330	<b>bne1</b>	L3	<b>01-1010-00-01-00</b>
	<b>0125</b>	1331		חוובג של L3	<b>0000000000-01</b>
END:	<b>0126</b>	1332	<b>stop1</b>		<b>00-1111-00-00-00</b>
STR:	<b>0127</b>	1333	.string	“abcdef”	<b>000001100001</b>
	<b>0128</b>	2000			<b>000001100010</b>
	<b>0129</b>	2001			<b>000001100011</b>
	<b>0130</b>	2002			<b>000001100100</b>
	<b>0131</b>	2003			<b>000001100101</b>
	<b>0132</b>	2010			<b>000001100110</b>
	<b>0133</b>	2011			<b>000000000000</b>
LENGTH:	<b>0134</b>	2012	.data	6,-9,15	<b>000000000110</b>
	<b>0135</b>	2013			<b>111111110111</b>
	<b>0136</b>	2020			<b>000000001111</b>
K:	<b>0137</b>	2021	.data	2	<b>000000000010</b>

כלומר תוכן קובץ ps.o הוא:

#### Base 4 Address                      Base 4 machine code

	123    23
1210	<b>200110</b>
1211	<b>020212</b>
1212	<b>000001</b>
1213	<b>200110</b>
1220	<b>020212</b>
1221	<b>000001</b>
1222	<b>202310</b>
1223	<b>010000</b>
1230	<b>013332</b>
1231	<b>200330</b>
1232	<b>010100</b>
1233	<b>121010</b>
1300	<b>000001</b>
1301	<b>130000</b>
1302	<b>333230</b>
1303	<b>203030</b>
1310	<b>333230</b>
1311	<b>000100</b>
1312	<b>113030</b>
1313	<b>000000</b>
1320	<b>000330</b>
1321	<b>000030</b>
1322	<b>000330</b>
1323	<b>000030</b>
1330	<b>122010</b>
1331	<b>000001</b>
1332	<b>033000</b>
1333	<b>001201</b>
2000	<b>001202</b>
2001	<b>001203</b>
2002	<b>001210</b>
2003	<b>001211</b>

2010	<b>001212</b>
2011	<b>000000</b>
2012	<b>000012</b>
2013	<b>333313</b>
2020	<b>000033</b>
2021	<b>000002</b>

LOOP        1233  
LENGTH      2012

קובץ ps.cent:

W        1212  
W        1221  
L3       1300  
L3       1331

קובץ ps.ext:

لتשומת לך : אם בקובץ מסוים אין הערה *extern*. אז לא יוצר עבورو קובץ *ext*. המותאים. כמו"ל עבורי קבצים שאינם מקרים הודיעות *entry*., במקרה זה לא יוצר קובץ *ent*. מותאים.

### סיכום והנחיות כלליות

- אורך התוכנית, הניתנת כקלט לאסטブル, אינו ידוע מראש (ואינו קשור לנגדל 1000 – של הזיכרון במעבד הדמיוני). ולכן אורךה של התוכנית המתורגם, אינו אמר להיות צפוי מראש. אולם בכך להקל בימוש התכנית, ניתן להניחס גודל מקטימלי. לפיק יש אפשרות להשתמש במערכות, לשם מטרה זו. במקרים אחרים בפרויקט, יש להשתמש על פי ייעילות/תוכנות נדרשות.
- קבצי הפלט של התוכנית, צריכים להיות בפורמט המופיע לעלה. שם של קבצי הפלט צריך להיות תואם לשם של תוכנית הקלט, פרט לסיומת. למשל, אם תוכנית הקלט היא *prog.as* או קבצי הפלט שייצרו הם : *prog.ob*, *prog.exl*, *prog.ent*.
- אופן הרצת התוכנית צריך להיות הואם לנדרש בממ"נ, ללא שינוים כלשהם. אין להוסיף תפריטים למיניהם וcdcומה. הפעלת התוכנית תעשה רק ע"י ארגומנטים של שורת פקודה.
- יש להקפיד לחלק את התוכנית למודולים. אין לרכז מספר מטרות במוחול יחיד. מומלץ לחלק למודולים כגון : מבני נתונים, מעבר ראשון, מעבר שני, טבלת סמלים וcdcומה.
- יש להקפיד ולתעד את הקוד, בצוירה מלאה וברורה.
- יש להקפיד על התעלומות מרוחים, ולהיות סלחניים כלפי תוכניות קלט, העושות שימוש ביותר רוחחים מהנדרש. למשל, אם לפקודה ישנו שני אופרנדים המופרדים בפסיק, אויל לפני שם הפוקודה או לאחר האופרנד הראשון או לאחר הפסק, יכול להיות מספר דוחות כלשהו, ובכל המקרים זו צריכה להיחשב פוקודה חוקית (פחות מבחינת הרוחחים).
- במקרה של תוכנית קלט, המכילה שגיאות תחביריות, נדרש להפיק, כמו באסטブルAMI, את כל השגיאות הקיימות, ולא לעזר לאחר התקולות בשגיאת הראשונה. כמון שעבור קובץ שאוי תחבירית, אין להפיק את קבצי הפלט (*ob*, *ext*, *ent*) אלא רק לדוח על השגיאות שנמצאו .

גם ונשלם חלק החסברים והגדירות הפרוייקט.

בשאלות ניתן לפנות אל :

קבוצת הדיון באתר הקורס, לכל אחד מהמנחים בשעות הקבלה שלהם. להזיכרכם, באפשרותו של כל סטודנט לפנות לכל מנהה, לאו דווקא למנחה הקבוצה שלו לקבל עזרה. שוב מומלץ בכל אלה מכם שטרם בדקו את האתר הקורס, לעשות זאת. נשאלות באתר זה הרבה שאלות בנושא חומר הלימוד והממן"טים, והתשובות יכולות לעזור לכם. לשומת לבכם, לא יתקבלו ממ"נים באיחור ללא סיבה מוצדקת, באישור מרכז הקורס.

בצלחה!!!