

# מטלת מנהה (ממ"ז) 14

הקורס : 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטרת הוגשה : פרויקט גמר

משקל המטרת הוגשה : 61 נקודות (חוובה)

מספר השאלות : 1

מועד אחרון להגשה : 14.3.2021

סמסטר : 2021א'

## קיימות שתי חלופות להגשת מטלות:

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס
- שליחת מטלות באמצעות דואר אלקטרוני - באישור המנהה בלבד

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

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

עליכם לכתוב תוכנת אSEMBLER, עברו שפת אSEMBLER שתוגדר בהמשך. הפרויקט יכתב בשפה C.

עליכם להגיש את הפריטים הבאים :

1. קבצי המקור של התוכנית שתכתבם (קבצים בעלי סימות c או h).
2. קובץ הרצה (מקומפל ומקושר) עברו מערכת אוביונטו.
3. קובץ makefile. הקימפול חייב להיות עם הקומפיילר gcc והדגלים : -Wall -ansi -pedantic יש לנפות את כל ההודעות שמוציאה הקומפיילר, כך שהתוכנית תתקמפל ללא כל העוראות או זהירות.
4. דוגמאות הרצה (קלט ופלט) :
  - א. קבצי קלט בשפה אSEMBLER, וקבצי הפלט שנוצרו מהפעלת האSEMBLER על קבצי קלט אלה. יש להציג שימוש במגוון הפעולות וטיפוסי הנתונים של שפת האSEMBLER.
  - ב. קבצי קלט בשפה אSEMBLER המציגים מגוון רחב של סוגי שגיאות אSEMBLER (ולכן לא נוצרים קבצי פלט), ותדפסי המסקן המראים את הודעות השגיאה שמוציאה האSEMBLER.

בשל גודל הפרויקט, עליכם לחלק את התוכנית במספר קבצי מקור, לפי משימות. יש להקפיד שקובץ המקור של התוכנית יעמוד בקריטריונים של בהירות, קריאות ו כתיבה נאה ומובנית.

זכור מספר היבטים חשובים של כתיבת קוד טוב :

1. הפשטה של מבני הנתונים : רצוי (כל האפשר) להפריד בין הגישה למבנה הנתונים בין השימוש של מבני הנתונים. כך, למשל, בעת כתיבת פונקציות לטיפול בטבלה, אין זה מעניינים של משתמשים בפונקציות אלה, האם הטבלה ממומשת באמצעות מערך או באמצעות רשימה מקוורת.
2. קריאות הקוד : יש להשתמש בשמות משמעותיים למשתנים ופונקציות. יש לעורוך את הקוד באופן מסודר : הזוחות עקביות, שורות ריקות להפרדה בין קטעי קוד, וכו'.
3. תיעוד : יש להכניס בקבצי המקור תיעוד תמצתי וברור, שיסביר את תפקידה של כל פונקציה (באמצעות הערות כותרת לכל פונקציה). כמו כן יש להסביר את תפקידם של משתנים חשובים. כמו כן, יש להכניס העורות ברמת פירוט גבוהה בכל הקוד.

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

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

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

מומלץ לקרוא את הגדרת הפרויקט פעמי ראונה ברכז, לקבלת תמונה כללית לגבי הנדרש, ורק לאחר מכן לקרוא שוב בצורה מעמיקה יותר.

### **רקע כללי ומטרת הפרויקט**

כידוע, קיימות שפות תוכנות רבות, ומספר גדול של תוכניות, הכתובות בשפות שונות, עשויות לרוץ באותו מחשב עצמו. כיצד "מכיר" המחשב כל כך הרבה שפות? התשובה פשוטה: המחשב מכיר למעשה שפה אחת בלבד: הוראות ונתונים הכתובים בקוד ביארי. קוד זה מאוחסן בגוש בזיכרונו, ונראה כמו רצף של ספרות בינאריות. יחידת העיבוד המרכזית - היע"מ (CPU) - יודעת לפרק את הרץ הזה לקטעים קטנים בעלי משמעות: הוראות, מענים ונתונים.

למעשה, זיכרונו המחשב כולל הוא אוסף של סיביות, שנוהגים לראותן כמקובצות ליחידות בעלות אורך קבוע (בתים, מילימטרים). לא ניתן להבחין, בין שאינה מיומנת, בהבדל פיסי כלשהו בין אותו חלק בזיכרון שבו נמצאת תוכנית לבין שאר הזיכרונות.

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

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

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

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

תפקידו של האסambilר הוא לבנות קובץ המכיל קוד מכונה, מקובץ נתן של תוכנית הכתובת בשפת אסמבלי. זהו השלב הראשון במסלול אותו עברת התוכנית, עד לקבלת קוד המוכן לריצה על חומרת המחשב. השלבים הבאים הם קישור (linkage) וטעינה (loading), אך בהם לא נעסק במיל'ן זה.

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

**لتשומת לב**: בהסבירים הכלליים על אופן עבודה תוכנת האסambilר, תהיה מדי פעם התייחסות גם לעבודת שלבי הקישור והטעינה. התייחסויות אלה נועדו על מנת לאפשר לכם להבין את המשך תהליך העיבוד של הפלט של תוכנת האסambilר. אין לטעות: עליהם לכתב את תוכנית האסambilר בלבד. אין לכתב את תוכניות הקישור והטעינה!!!

## המחשב הדמיוני ושפת האסטREL

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

"חומרה" :

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

למעבד 8 אוגרים כלליים, בשמות : r0, r1, r2, r3, r4, r5, r6, r7. גודלו של כל אוגר הוא 12 סיביות. הסיבית ה-0 היא השם המשמעותי ביותר ביותר במס' 11. שמות האוגרים כתובים תמיד עם אות 'z', קטנה.

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

גודל הזיכרון הוא 4096 תאים, בכתובות 4095-0, וכל תא הוא בגודל של 12 סיביות. לתא בזיכרון נקרא גם בשם "מילה". הסיביות בכל מילה ממושפרות כמו באוגר.

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

מבנה הוראות המוכנה :

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

כל הוראה מוכנה מוקודדת במספר מילוט זיכרון רצופות, החל ממילה אחת ועד למקסימום שלוש מילים, בהתאם לסוג הפעולה (ראו פרטיהם בהמשך).

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

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

0	1	2	3	4	5	6	7	8	9	10	11
מייען יעד	מייען מקור	func	opcode								

במודול המוכנה שלנו יש 16 פעולות, בפועל, למגוון שימושים לקודד יותר פעולות. כל פעולה מיוצגת בשפת אסטREL באמצעות סימболים על ידי **שם-פעולה**, ובקוד המוכנה על ידי קומבינציה ייחודית של ערכי שני שדות במילה הראשונה של ההוראה : **קוד-הפעולה (opcode)**, **פונקציה (funct)**.

להלן טבלת הפעולות :

שם הפעולה	(בבסיס עשרוני)	funcf (בבסיס עשרוני)	opcode (בבסיס עשרוני)
mov			0
cmp			1
add	10		2
sub	11		2
lea			4
clr	10		5
not	11		5
inc	12		5
dec	13		5
jmp	10		9
bne	11		9
jsr	12		9
red			12
prn			13
rts			14
stop			15

הערה : שם-הפעולה נכתב תמיד באותיות קטנות. פרטים על מהות הפעולות השונות יובאו בהמשך.

להלן מפרט השדות בamilha הראשונה בקוד המכוונה של כל הוראה.

**סיביות 8-11:** סיביות אלה מכילות את קוד-הפעולה (opcode). ישן מספר פעולות עם קוד פעולה זהה (ראו בטבלה לעיל, קוד-פעולה 2, 5 או 9), ומה שבדיל ביניהן הוא השדה funct.

**סיביות 4-7:** שדה זה, הנקרא funct, מתייחס כאשר מדובר בפעולת שקוד-הפעולה (opcode) שלה משותף לכמה פעולות שונות (כאמור, קוד-פעולה 2, 5 או 9). השדה funct יכול ערך ייחודי לכל פעולה מקובצת הפעולות שיש להו אותו קוד-פעולה. אם קוד-הפעולה משתמש בפעולת אחת בלבד, הסיביות של השדה funct יהיו מאופסות.

**סיביות 2-3:** מכילות את מספירה של שיטת המיעון של אופרנד המקור. אם אין בהוראה אופרנד מקור, סיביות אלה יהיו מאופסות. מפרט של שיטות המיעון השונות ניתן בהמשך.

**סיביות 0-1:** מכילות את מספירה של שיטת המיעון של אופרנד היעד. אם אין בהוראה אופרנד יעד, סיביות אלה יהיו מאופסות.

שיטות מיעון :

שיטות מיעון (addressing modes) הן האופנים השונים בהם ניתן להעביר אופרנדים של הוראות מכונה. בשפת האסמלבי שלנו קיימות ארבע שיטות מיעון, המסומנות במספרים 0,1,2,3.

השימוש בשיטות המיעון מצריך מילוט-מידע נוספת בקוד המכוונה של הוראה, בנוסף למילה הראשונה. לכל אופרנד של הוראה נדרש מילת-מידע אחת נוספת. כאשר בהוראה יש שני אופרנדים, קודם תופיע מילת-המידע של אופרנד המקור, ולאחריה מילת-המידע של אופרנד היעד. להלן המפרט של שיטות המיעון.

מספר	שיטת המיעון	תוכן מילת-המידע הנוספת	תחביר האופרנד באסמלבי	דוגמה
0	מיון מיידי (immediate)	AMILT-MIDU NOSFAT SHL HAHORAH MCILAH AT HAOPERND UZMO, SHHOA MASFER SHLEM BBASIS MASHLIM L-2, BROCHB SHL 12 SIVIOT	האופרנד מתחיל בטע # ולאחריו ובצמוד אליו מופיע מספר שלם בסיסי עשרוני.	mov #1, r2  בדוגמה זו האופרנד הראשון של ההוראה (אופרנד המקור) נתון בשיטת מיון מיידי. ההוראה כתובת את הערך 1- אל אוגר r2.
1	מיון ישיר (direct)	AMILT-MIDU NOSFAT SHL HAHORAH MCILAH CTOBAT BIZIKRON. HAMILAH BCCTOBAT ZO BIZIKRON HIA HAOPERND. HCTOBAT MIYICHTA CMSFER <u>LALA SIMON</u> BROCHB SHL 12 SIVIOT.	האופרנד הוא <u>תווית</u> שכבר הגדרה, או שתוגדר בהמשך הקובץ. ההגדרה נעשית על ידי כתיבת התווית בתחילת השורה של הנחית 'data'. או של הוראה, או באמצעות אופרנד של הנחית 'string'. התווית מייצגת באופן סימבולי כתובות בזיכרון.  <u>דוגמה נוספת:</u> ההוראה jmp next  מבצעת קפיצה אל השורה בה מוגדרת התווית next (כלומר ההוראה הבאה שתבוצע נמצאת כתובות next).  התווית next מוגדרת בAMILT-HMIDU NOSFAT.	השורה הבאה מוגדרה את התווית x : x: .data 23  ההוראה : dec x  מקטינה ב-1 את תוכן המילה שבכתובת x בזיכרון (ה"משתנה" x). הכתובת x מוגדרת בAMILT-HMIDU NOSFAT.  <u>דוגמה נוספת:</u> ההוראה jmp next  מבצעת קפיצה אל השורה בה מוגדרת התווית next (כלומר ההוראה הבאה שתבוצע נמצאת כתובות next).  התווית next מוגדרת בAMILT-HMIDU NOSFAT.
2	מיון יחסי (relative)	SHIETA ZO RLOONETIAT <u>AD ORK</u> LHHORAOOT HAMBUTZOT KFICHA (HSTAVUPOT) LHHORAH ACHART. MDOBER BHORAOOT UM KOD-P'ULAH 9 BLBED : jmp, bne, jsr LA NIYUN LEHSHTEM SHIETA ZO BHORAOOT UM KODI-P'ULAH ACHARIM.  BISHIETA ZO, YIS BKIDUD HAHORAH AMILT MIDU NOSFAT, MCILAH AT MRACH KKFICHA, BMILLOT ZICKRON, MMILT-HMIDU NOSFAT HANOCHEIT AL HMILAH HARESHONA SHL HAHORAH HMVKOSHET (HHORAH HABA LBIVTU).	האופרנד מתחיל בטע % ולאחריו ובצמוד אליו מופיע שם של תווית.  התווית מייצגת באופן סימבולי כתובות של הוראה <u>בקובץ המקור הנוכחי של</u> <u>התוכנית.</u>  YITCAN SHHTOVIIT CABR HOGDRAH, AO SHTOGDAR BEMASHK KCOVTS. HHGDRHA NEUSHIT UL IDI CTIBAH HTTOVIIT BTOCHLIT SHORT HAHORAH.	jmp %next  בדוגמה זו, ההוראה jmp מבצעת קפיצה אל השורה בماה מוגדרת התווית next במה מוגדרת התווית next (כלומר ההוראה הבאה שתבוצע נמצאת כתובות next).  נניח, לדוגמה, כי ההוראה kmpl שבדוגמה נמצאת בכתובת 500 (עשרוני). כמו כן, נניח כי התווית next מוגדרת בקובץ המקור הנוכחי בכתובת 300 (עשרוני).  מרחיק הקפיצה ממילט- המידע של ההוראה jmp אל הכתובת next הוא 300-(500+1)=201 ולכן המילה הנוספת תכיל את הערך -201.

מספר	שיטת המיעון	תוכן מילת-המידע הנוספת	תחביר האופרנד באסמלבי	דוגמה
3	מיון אוגר ישיר (register direct)	האוגר הוא אוגר. AMILT-MIDU NOSFAT SHL HORA MCILAH BSIYIOT 7-0 BIT DOLK YICHID HAMYICIG AT HAOGER HMTAIM. SIYIOT 0 TDLOK AM MZDBR BAOGER 0Z, SIYIOT 1 TDLOK AM MZDBR BAOGER 1Z VCO. SIYIOT 11-8 YH TMID MAOPSFOT	האופרנד הוא שם של אוגר.	clr r1  בדוגמה זו, ההוראה מפסיק את האוגר r1. המילה הנוספת של ההוראה תכיל (בבינארית) 0000000000010  <u>דוגמה נוספת:</u> mov #1, r2  האופרנד השני של ההוראה (אופרנד היעד) נתון בשיטת מיון אוגר ישיר. ההוראה כתובת ישר. ההוראה כותבת את הערך המיידי 1- אל אוגר r2. המילה הנוספת השנייה של ההוראה תכיל (בבינארית) 000000000100

#### מפורט הוראות המכונה:

בתיאור הוראות המכונה נשתמש במונח **PC** (קייזר של "Program Counter"). זהו אוגר פנימי של המעבד (לא אוגר כללי), שמכיל בכל רגע נתון את כתובת הזיכרון בה נמצאת ההוראה **הקיימת שמתבצעת** (הכוונה תמיד לכתובת המילה הראשונה של ההוראה).

horאות המכונה מתחולקות לשולש קבוצות, לפי מספר האופרנדים הנדרשים לפעולה.

#### **קבוצת ההוראות הראשונה:**

אליהן הוראות המקבלות שני אופרנדים.

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

הוראה	opcode	funct	הפעולה המתבצעת	דוגמאות	הסבר הדוגמה
mov	0		מבצע העתקה של תוכן אופרנד המקור (האופרנד הראשון) אל אופרנד היעד (האופרנד השני).	mov A, r1	העתק את תוכן המשתנה A (המילה שבכ坨ות בזיכרון) אל אוגר r1.
cmp	1		מבצע השוואת בין שני האופרנדים. ערך אופרנד היעד (השני) מופחת מערך אופרנד המקור (הראשון), ללא שמירת תוצאת החישוב מעדכנת דגל בשם Z ("דגל האפס") באוגר הסטטוס (PSW).	cmp A, r1	אם תוכן המשתנה A זהה לתוכנו של אוגר r1 אז הדגל Z ("דגל האפס") באוגר הסטטוס (PSW) יודלק, אחרת הדגל יאפס.
add	2	10	אופרנד היעד (השני) מקבל את תוצאה החיבור של אופרנד המקור (הראשון) והיעד (השני).	add A, r0	אוגר 0Z מקבל את תוצאה החיבור של תוכן המשתנה A ותוכנו הנוכחי של 0Z.
sub	2	11	אופרנד היעד (השני) מקבל את תוצאה החישוב של אופרנד המקור (הראשון) מואופרנד היעד (השני).	sub #3, r1	אוגר 1Z מקבל את תוצאה החישוב של הקבוע 3 מתוכנו הנוכחי של האוגר r1.
lea	4		lea הוא קיזר (ראשי תיבות) של load effective address מציבה את מען הזיכרון המזיג על ידי התווית שבאופרנד הראשון (המקור), אל האופרנד השני (היעד).	lea HELLO, r1	המען שמייצגת התווית HELLO מוצב לאוגר r1.

### קבוצת ההוראות השנייה:

אלן הוראות המקבלות אופרנד אחד בלבד. אופן הקידוד של האופרנד הוא כמו של אופרנד היעד בהוראה עם שני אופרנדים. השדה של אופרנד המקור (סיביות 2-3) בamilha הראשונה בקידוד ההוראה אינם בשימוש, ולפיכך יהיו מאופסים.

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

הוראה	opcode	funct	הפעולה המתבצעת	דוגמה	הסביר הדוגמה
clr	5	10	איפוס תוכן האופרנד.	clr r2	האגור 2 ז' מקבל את הערך 0.
not	5	11	היפוך הסיביות באופרנד (כל סיבית שערכה 0 תהפוך ל-1 ולהיפך : 1 ל-0).	not r2	כל בית באגור 2 ז' מתחפה.
inc	5	12	הגדלת תוכן האופרנד באחד.	inc r2	תוכן האגור 2 ז' מוגדל ב-1.
dec	5	13	הקטנת תוכן האופרנד באחד.	dec Count	תוכן המשתנה Count מוקטן ב-1.
jmp	9	10	קפיצה (הסתעפות) בלתי מותנית אל הוראה שנמצאת במען המיוצג על ידי האופרנד. ככלומר, כתוצאה מביצוע ההוראה, מצביע התוכנית (PC) מקבל את כתובת יעד הקפיצה.	jmp %Line	בשיטת מיון יחסי, המרחק לתוכית Line מהתווסף למצויע התוכנית ולפיכך ההוראה הבאה שתתבצע תהיה במען Line.
bne	9	11	asm ערך הדגל Z באגור הסטטוס (PSW) הוא 0, אזי PC ← address(Line) מצביע התוכנית קיבל את כתובת התוויות Line, ולפיכך ההוראה הבאה שתתבצע תהיה במען Line.	bne Line	asm קיצור (ראשי תיבות) של branch if not equal (to zero). זהה הוראות הסתעפות מותנית. אם ערכו של הדגל Z באגור הסטטוס (PSW) הינו 0, אזי מצביע התוכנית (PC) מקבל את כתובת יעד הקפיצה. כזכור, הדגל Z נקבע באמצעות הוראת cmp.
jsr	9	12	קריאה לשגרה (סברוטינה). כתובת ההוראה שאחרי הוראות jsr הנווכית (PC+2) נדחפת לתוך המחסנית שביברן המחשב, ומצויע התוכנית (PC) מקבל את כתובת השגרה. הערה: חוזרת מהשגרה מתבצעת באמצעות הוראות dzr, תוך שימוש בכתובת שבמחסנית.	jsr SUBR	push(PC+2) PC ← address(SUBR) מצביע התוכנית קיבל את כתובת התוויות SUBR, ולבסוף, ההוראה הבאה שתתבצע SUBR. SUBR. כתובת החזרה מהשגרה נשמרת במחסנית.
red	12		קריאה שלתו מהקלט הסטנדרטי (stdin) אל האופרנד.	red r1	קוד הא-ascii של התו הנקרא מהקלט ייכנס לאגור r1.
prn	13		הדפסת התו הנמצא באופרנד, אל הפלט הסטנדרטי (stdout).	prn r1	יודפס לפט התו (קוד ascii r1 הנמצא באגור r1)

### קבוצת ההוראות השלישייה:

אלן הוראות ללא אופרנדים. קידוד ההוראה מורכב ממיליה אחת בלבד. השדות של אופרנד המקור ושל אופרנד היעד (סיביות 0-3) בamilha הראשונה של ההוראה אינם בשימוש, ולפיכך יהיו מאופסים.

ההוראות השייכות לקבוצה זו הן: rts, stop

הוראה	opcode	funct	הפעולה המתבצעת	דוגמה	הסביר הדוגמה
rts	14		מתבצעת חזרה משגרה. הערך שבראש המחסנית של המחשב מזחצן מן המחסנית, ומוכנס למצביע התוכנית (PC).	rts	PC ← pop() ההוראה הבאה שתתבצע תהיה זו שאחרי הוראת jsr שקרה לשגרה.
stop	15		歇止。即停止执行。通常用于程序结束。	stop	התוכנית עצצת התוכנית.

## מבנה תכנית בשפת אסמבלי:

תכנית בשפת אסמבלי בנויה ממשפטים (statements). קובץ מקור בשפת אסמבלי מורכב משורות המכילות ממשפטים של השפה, כאשר כל ממשפט מופיע בשורה נפרדת. כלומר, הפרדה בין ממשפט למשפט בקובץ המקור הינה באמצעות התו 'ז' (שורה חדשה).

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

יש ארבעה סוגי משפטיים (שורות בקובץ המקור) בשפת אסמבלי, וهم :

סוג המשפט	הסבר כללי
משפט ריק	זהי שורה המכילה אך ורק תווים לבנים (whitespace), ככלומר רק את התווים ' ' ו- 't' (רווחים וטאים). ייתכן ובסורה אין אף תו (למעט התו '\n), ככלומר השורה ריקה.
משפט הערת	זהי שורה בה התו הראשוני הינו ';' (נקודה פסיק). על האסמבלי להתעלם לחולטן משורה זו.
משפט הנטהיה	זהו משפט המנחה את האסמבלי מה עליו לעשות כשהוא פועל על תוכנית המקור. יש מספר סוגים של משפטי הנטהיה. משפט הנטהיה עשוי לגרום להקצת זיכרונו ואתחול משתנים של התוכנית, אך הוא אינו מייצר קידוד של הוראות מכונה המיועדות לביצוע בעת ריצת התוכנית.
משפט הוראה	זהו משפט המיציר קידוד של הוראות מכונה לביצוע בעת ריצת התוכנית. המשפט מורכב ממשם ההוראה (פעולה) שעל המעבד לבצע, והאופרנדים של ההוראה.

cut נפרט יותר לגבי סוגי המשפטיים השונים.

### משפט הנטהיה:

משפט הנטהיה הוא בעל המבנה הבא :

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

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

יש ארבעה סוגיים (שמות) של משפטי הנטהיה, וهم :

1. הנטהיה 'data'.

הפרמטרים של הנטהיה '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 לפסק ובין פסיק למספר יכולים להופיע רווחים וטאים בכל כמות (או בכלל לא), אולם הפסיק תיבר 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 האסמבלי להקצות מקומות בתמונה הנתונים (data image), אשר בו יוחסנו הערךים של הפרמטרים, ולקדם את מונה הנתונים, בהתאם למספר הערכים. אם בהנחת data. מוגדרת תווית, אז תווית זו מקבלת את ערך מונה הנתונים (לפני הקידום),

ומוכנסת אל טבלת הסמלים. דבר זה מאפשר להתייחס אל מקום מסוים בתמונה הנתונים דרך שם התוויות (למעשה, זהה דרך להגדיר שם של משתנה).

כלומר אם נכתב :

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

אזי יוקצו בתמונה הנתונים ארבע מיללים רצופות שיכילו את המספרים שמופיעים בהנחיה. התוויות XYZ מזוהה עם כתובות המילה הראשונה.

אם נכתב בתוכנית את הוראה :  
mov XYZ, r1

אזי בזמן ריצת התוכנית יוכנס לאוגר r1 ערך 7.

ואילו הוראה :

lea XYZ, r1

תכנס לאוגר r1 את ערך התוויות XYZ (כלומר הכתובת בזיכרון בה מאוחסן הערך 7).

2. ההנחיה '.string'

להנחיה '.string', פרמטר אחד, שהוא מחרוזת חוקית. תווי המחרוזת מקודדים לפי ערכי ascii-ה-המתאים, ומוכנסים אל תמונה הנתונים לפי סדרם, כל تو במילה נפרדת. בסוף המחרוזת יתווסף התו '0' (ערך המספרי 0), המסמן את סוף המחרוזת. מונה הנתונים של האסטבלר יקודם בהתאם לאורך המחרוזת (בתוספת מקום אחד עבור התו המסיים). אם בשורת ההנחיה מוגדרת תוויות, אזי תוויות זו מקבלת את ערך מונה הנתונים (לפניהם) ומוכנסת אל טבלת הסמלים, בדומה כפי שנעשה עבור '.data'. (כלומר ערך התוויות יהיה הכתובת בזיכרון שבה מתחליה המחרוזת).

לדוגמה, ההנחיה :

STR: .string "abcdef"

מקצת בתמונה הנתונים רצף של 7 מיללים, ומאתחלת את המיללים לקודי ascii של התווים לפי הסדר במחרוזת, ולאחריהם הערך 0 לסימון סוף מחרוזת. התוויות STR מזוהה עם כתובות התחלה המחרוזת.

3. ההנחיה '.entry'

להנחיה '.entry', פרמטר אחד, והוא שם של תוויות המוגדרת בקובץ המקור הנוכחי (כלומר תוויות שמקבלת את ערכיה בקובץ זה). מטרת ההנחיה entry. היא לאפיין את התוויות הזו באופן שיאפשר לקוד אסטבלר הנמצא בקובץ מקור אחרים להשתמש בה (כאופrnd של הוראה).

לדוגמה, השורות :

HELLO: .entry HELLO  
add #1, r1

מודיעות לאסטבלר שאפשר להתייחס בקובץ אחר לתוויות HELLO המוגדרת בקובץ הנוכחי.

לתשומתך : תוויות המוגדרת בתחילת שורת entry. הינה חסרת משמעות והאסטבלר מתעלם מהתוויות זו (אפשר שאסטבלר יוציא הודעה אחרת).

4. ההנחיה '.extern'

להנחיה '.extern', פרמטר אחד, והוא שם של תוויות שאינה מוגדרת בקובץ המקור הנוכחי. מטרת ההוראה היא להודיע לאסטבלר כי התוויות מוגדרת בקובץ אחר, וכי קוד האסטבלר בקובץ הנוכחי עושה בתוויות שימוש.

נשים לב כי הנקיה זו תואמת להנchia 'entry.' המופיע בקובץ בו מוגדרת התווiot. בשלב הקישור תבוצע התאמה בין ערך התווiot, כפי שנקבע בקובץ המכוון של הקובץ שהגדיר את התווiot, לבין קידוד ההוראות המשמשות בתווiot בקבצים אחרים (שלב הקישור אינו רלוונטי לממ'ן זה).

לדוגמא, משפט ההנchia 'extern.' התואם למשפט ההנchia 'entry.' מהדוגמה הקודמת יהיה:

.extern HELLO

הערה: לא ניתן להגדיר באותו הקובץ את אותה התווiot גם `c-entry` וגם `c-extern` (בדוגמאות לעיל, התווiot `HELLO`).

להשומת לב: תווiot המוגדרת בתחילת שורת `extern`. הינה חסרת משמעות והאסמבלר **מתעלם** מהתווiot זו (אפשר שהאסמבלר יוציא הודעה אחרת).

#### משפט הוראה:

משפט הוראה מורכב מחלקים הבאים:

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

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

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

כאשר יש שני אופרנדים, האופרנדים מופרדים זה מזה ב' , ' (פסיק). בדומה להנchia 'data.', לא **חייבת להיות הצמדה של האופרנדים לפסיק**. כל כמות של רווחים ו/או ט-abs משיוני צידי הפסיק היא חוקית.

למשפט הוראה עם שני אופרנדים המבנה הבא:

label: opcode source-operand, target-operand

לדוגמא:

HELLO: add r7, B

למשפט הוראה עם אופרנד אחד המבנה הבא:

label: opcode target-operand

לדוגמא:

HELLO: bne %XYZ

למשפט הוראה ללא אופרנדים המבנה הבא:

label: opcode

לדוגמא:

END: stop

#### אפיון השדות במשפטים של שפת האסמבלי

תווiot:

תווiot היא סמל שמודדר בתחילת שפט הוראה' או בתחילת הנchia `data`. או `string`.  
תווiot חוקית מתחילה באות אלפביתית (גדולה או קטנה), ולאחריה סדרה כלשהי של אותיות אלפביתיות (גדולות או קטנות) ו/או ספרות. האורך המקסימלי של תווiot הוא 31 תווים.

הגדירה של תווית מסוימת בתו : (נקודותים). זו זה אינו מהו חלק מהתוויות, אלא רק סימן המציין את סוף ההגדירה. התו : חייב להיות צמוד לתווית (ללא רווחים).

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

לדוגמא, התוויות המוגדרות להלן הן תוויות חוקיות.

hEllo:

x:

He78902:

**لتשומת לב :** מילים שמורות של שפט האסמבלי (כלומר שם של פעולה או הינה, או שם של אוגר) אינן יכולות לשמש גם כשם של תווית. לדוגמה : הסמלים add, z לא יכולים לשמש כתוויות, אבל הסמלים Add, R, R3 הם תוויות חוקיות.

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

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

מספר :

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

מחירות :

מחירות חוקית היא סדרת תווים ascii נראים (שניתנים להדפסה), המוקפים במרכאות כפولات (המרכאות אינן נחשבות חלק מהמחירות). דוגמה למחירות חוקית : “hello world”.

### סימון המילים בקוד המכונה באמצעות המאפיין ”A,R,E”

האסמבילר בונה מלכתחילה קוד מכונה שמיועד לטעינה החל מctaובת 100. אולם, לא בכל פעם שהקוד יטען לזרק הרצה, מובטח שאפשר יהיה לטען אותו החל מctaובת 100. במקרה כזה, קוד המכונה הנตอน אינו מותאים וכיורק לתקן אותו. לדוגמה, מילת-הມידע של אופrnd בשיטת מעון ישיר לא תהיה נcona, כי הכתובות השתנותה.

הרעיון הוא להכניס תיקונים בקוד המכונה בכל פעם שייטען לזרק הרצה. כך אפשר יהיה לטען את הקוד בכל פעם למקום אחר, בלי צורך לחזור על תהליך האסמבלי. תיקונים כאלה נעשים בשלב הקישור והטיענה של הקוד (אנו לא מטפלים בכך בממ'ן זה), אולם על האסמבילר להוסיף מידע בקוד המכונה שיאפשר לזהות את הנקודות בקוד בהן נדרש תיקון.

לצד כל מילה בקוד המכונה, האסמבילר מוסיף מאפיין שנקרא ”A,R,E”. לכל מילה בקוד, מוצמד שדה המכיל את אחת האותיות A או R או E.

- האות A (קייזר של Absolute) בא להזכיר שתוכן המילה אינו תלוי במקום בו זיכרונו בו ייטען בפועל קוד המכונה של התוכנית בעת ביצועה (למשל, המילה הראשונה בכל הוראה, או מילת-מידע המכילה אופrnd מיידי).
- האות R (קייזר של Relocatable) בא להזכיר שתוכן המילה תלוי במקומות בו זיכרונו בו ייטען בפועל קוד המכונה של התוכנית בעת ביצועה (למשל, מילת-מידע המכילה כתובות של תווית המוגדרת בקובץ המקור הנוכחי).
- האות E (קייזר של External) בא להזכיר שתוכן המילה תלוי בערכו של סמל שאינו מוגדר בקובץ המקור הנוכחי (למשל, מילת-מידע המכילה ערך של סמל המופיע בהנחיית `extern`).

**נשים לב כי רוב המילים בקוד המכונה מאופיינות על ידי האות A.** למעשה, רק מילת-המידע הנוספת של שיטת מיון ישר תואופיין על ידי האות R או E (תלוימ את האופrnd בקוד האסמבלי הוא תווית מקומית או סמל חיצוני).

### אסמבילר עם שני מעברים

כאשר מקבל האסמבילר כקלט תוכנית בשפת אסמבלי, עליו לעבור על התוכנית פעמיים. במעבר הראשון, יש לזהות את הסמלים (תוויות) המופיעים בתוכנית, ולתת לכל סמל ערך מספרי שהוא המعن בזיכרו שהסמל מיצג. במעבר השני, באמצעות ערכי הסמלים, וכן קוד-הפעלה ומספרים האוגרים, בונים את קוד המכונה.

לדוגמה : האסמבילר מקבל את התוכנית הבאה בשפת אסמבלי :

```

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

```

קוד המכונה של התוכנית (הוראות ונתונים) נבנה כך שיתאים לטעינה בזיכרון החל ממגען 100 (עשרהונи).

התרגום של תוכנית תכנית המקור שבדוגמה לקוד בינארי מוצג להלן :

Address (decimal)	Source Code	Explanation	Machine Code (binary)	"A,R,E"
0100	MAIN: add r3, LIST	First word of instruction	001010101101	A
0101		Register r3	000000001000	A
0102		Address of label LIST	000010000101	R
0103	LOOP: prn #48		110100000000	A
0104		Immediate value 48	000000110000	A
0105	lea STR, r6		010000000111	A
0106		Address of label STR	000010000000	R
0107		Register r6	000001000000	A
0108	inc r6		010111000011	A
0109		Register r6	000001000000	A
0110	mov r3, K		000000000101	A
0111		Register r3	0000000001000	A
0112		Address of label K	000010001000	R
0113	sub r1, r4		001010111111	A
0114		Register r1	0000000000010	A
0115		Register r4	000000010000	A
0116	bne END		100110110001	A
0117		Address of label END	000001111111	R
0118	cmp vall, #-6		000100000100	A
0119		Address of extern label vall	0000000000000	E
0120		Immediate value -6	111111111010	A

Address (decimal)	Source Code	Explanation	Machine Code (binary)	"A,R,E"
0121	bne %END		100110110010	A
0122		Distance to label END	000000000101	A
0123	dec K		010111010001	A
0124		Address of label K	000010001000	R
0125	jmp %LOOP		100110100010	A
0126		Distance to label LOOP	111111101001	A
0127	END: stop		111100000000	A
0128	STR: .string "abcd"	Ascii code 'a'	000001100001	A
0129		Ascii code 'b'	000001100010	A
0130		Ascii code 'c'	000001100011	A
0131		Ascii code 'd'	000001100100	A
0132		Ascii code '\0'	000000000000	A
0133	LIST: .data 6, -9	Integer 6	0000000000110	A
0134		Integer -9	111111101111	A
0135	.data -100	Integer -100	111110011100	A
0136	K: .data 31	Integer 31	000000011111	A

האסמבלר מחזיק טבלה שבה רשומים כל שמות הפעולה של הוראות והקודים הבינאריים (opcode, funct) המותאימים להם, וכן שמות הפעולות ניתנים להמרה לבינארי בקלות. כאשר נקרא שם פעולה, אפשר פשוט לעיין בטבלה ולמצוא את הקוד הבינארי.

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

למשל, בקוד לעיל, האסמבלר אינו יכול לדעת שהסמל END אמור להיות משוייך למן 127 (עשרוני), והסמל K אמור להיות משוייך למן 136, אלא רק לאחר שנקרוו כל שורות התוכנית.

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

במעבר הראשון נבנית טבלת סמלים בזיכרון, ובה לכל סמל שבתוכנית המקור משוייך ערך מסווני, שהוא ממן בזיכרון.

במעבר השני נעשית ההמרה של קוד המקור לקוד מכונה. בתחילת המעבר השני צריכים הערכים של כל הסמלים להיות כבר ידועים

בעור הדוגמה, טבלת הסמלים נתונה להלן. לכל סמל יש בטבלה גם מאפיינים (attributes) שיוסברו בהמשך. אין חשיבות לסדר השורות בטבלה (כאן הטבלה לפי הסדר בו הוגדרו הסמלים בתוכנית).

Symbol	Value (decimal)	Attributes
MAIN	100	code
LOOP	103	code
END	127	code
STR	128	data
LIST	133	data
K	136	data, entry
val1	0	external

لتשומת לב: תפקיד האסמבלר, על שני המעברים שלו, לתרגם קובץ מקור לשפת מכונה. בגמר פעולות האסמבלר, התוכנית טרם מוכנה לטיעינה לזכרון לצורך ביצוע. קוד המכונה חייב לעמוד בשלבי הקישור/טיענה, ורק לאחר מכן לשלב הביצוע (שלבים אלה אינם חלק מהמשמעות).

## המעבר הראשון

במעבר הראשון נדרשים כללים כדי לקבוע איזה מען ישוק לכל סמל. העיקרונו הבסיסי הוא לספור את המקומות בזיכרוון, אותן תפוזות ההוראות. אם כל הוראה תיטען בזיכרוון למקום העוקב להוראה הקודמת, תציגו ספירה כזאת את מען ההוראה הבאה. הספירה נעשית על ידי האסטבלר ומוחזקת במונה ההוראות (IC). ערכו ההתחלתי של IC הוא 100 (עשורוני), ולכן IC מוגדל בכל שורת המכונה של ההוראה הראשונה נבנה כך שייטען בזיכרוון החל ממען 100. ה-IC מוגדל במספר ההוראה המקצת מקום בזיכרוון. לאחר שהאסטבלר קובע מהו אורך ההוראה, ה-IC מוגדל במספר התאים (מיילים) הנתפסים על ידי ההוראה, וכך הוא מצביע על התא הפנוי הבא.

כאמור, כדי לקודד את ההוראות בשפת מכונה, מוחזק האסטבלר טבלה, שיש בה קידוד מתאים לכל שם פעולה. בזמן התרגום מחליף האסטבלר כל שם פעולה בקידוד שלו. כמו כן, כל אופרנד מוחלף בקידוד מתאים, אך פעולה ה恰恰פה זו אינה כה פשוטה. ההוראות משתמשות בשיטות מיון מגוונות לאופרנדים. אותה פעולה יכולה לקבל משמעותות שונות, בכל אחת משיטות המיון, וכן יתאים לה קידודים שונים לפי שיטות המיון. לדוגמה, פעולה ההזזה屯 יכולת להתייחס להעתיקת תוכן תא זיכרוון לאוגר, או להעתיקת תוכן אוגר לאוגר אחר, וכן הלאה. ככל אפשרות כזו屯 עשוי להיות להתאים קידוד שונה.

על האסטבלר לסרוק את שורת ההוראה בשלמותה, ולהחליט לגבי הקידוד לפי האופרנדים. בדרך כלל מתחלק הקידוד לשדה של שם הפעולה, ושדות נוספים המכילים מידע לגבי שיטות המיון. כל השדותividually ימלה אחת או יותר בקוד המכונה.

כאשר נתקל האסטבלר בתוויות המופיעות בתחילת השורה, הוא יודע שלפניו הגדרה של תווית, ואז הוא משייך לה מען – תוכנו הנוכחי של IC. כך מקבלות כל התוויות את מענייהן בעת ההגדרה. תוויות אלה מוכנסות לטבלת הסמלים, המכילה בנוסף לשם התווית גם את המען ומאפיינים נוספים. כאשר תהיה התיקשות לתוויות באופרנד של הוראה כלשהי, יוכל האסטבלר לשולף את המען המתאים לטבלת הסמלים.

הוראה יכולה להתיחס גם לסמל שטרם הוגדר עד כה בתוכנית, אלא יוגדר רק בהמשך התוכנית. להלן, לדוגמה, הוראת הסתעפות למען שМОוגדר על ידי התווית A שמשמעותו רק בהמשך הקוד:

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

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

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

## המעבר השני

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

לשם כך מבצע האסטבלר מעבר נוסף (מעבר שני) על כל קובץ המקור, ומעדכן את קוד המכונה של האופרנדים המשמשים בסמלים, באמצעות ערכי הסמלים בטבלת הסמלים. בסוף המעבר השני, תהיה התוכנית מתורגמת בשלמותה לקוד מכונה.

## הפרדת הוראות ונתונים

בתוכנית מבחינים שני סוגי של תוכן: הוראות ונתונים. יש לארון את קוד המוכנה כך שתהייה הפרדה בין הנתונים וההוראות. הפרדת ההוראות והנתונים לקטעים שונים בזיכרון היא שיטה עדיפה על פני הצמדה של הגדרות הנתונים להוראות המשמשות בהן.

אחד הסכנות הטമונות באין הפרדת ההוראות מהנתונים היא, שלפעמים עלול המעבד, בעקבות שגיאה לוגית בתוכנית, לנסות "לבצע" את הנתונים כאילו היו הוראות חוקיות. למשל, שגיאה שיכולה לגרום לתופעה כזו היא הסטעפות לא נכון. התוכנית כמובן לא תעבור נכוון, אך לרוב הנזק הוא יותר חמוץ, כי נוצרת חריגת חומרה ברגע שהמעבד מבצע פעולה שאינה חוקית.

האסםבלר שלנו **חייב להפריד**, בקוד המוכנה שהוא מציר, בין קטע הנתונים לקטע ההוראות. **כלומר בקובץ הפלט (בקוד המוכנה) תהיה הפרדה של הוראות ונתונים לשני קטיעים נפרדים**, אם כי **בקובץ הקלט אין חובה שתהייה הפרדה כזו**. בהמשך מתואר אלגוריתם של האסםבלר, ובו פרטים כיצד לבצע את ההפרדה.

### גילי שגיאות בתוכנית המקור

האסםבלר אמר או לגנות ולדוח על שגיאות בתחביר של תוכנית המקור, כגון פעולה שאינה קיימת, מספר אופרנדים שגוי, סוג אופרנד שאינו מתאים לפעולה, שם אונגר לא קיים, ועוד שגיאות אחרות. כמו כן מודיא האסםבלר שככל סמל מוגדר פעם אחת בדוק.

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

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

**להשומת לב:** האסםבלר אין עוצר את פעולתו אחרי שנמצאה השגיאה הראשונה, אלא ממשיך לעبور על הקלט כדי לגנות שגיאות נוספות, ככל שישנן. כਮובן שאיין כל טעם לייצר את קבצי הפלט אם נתגלו שגיאות (ممילא אי אפשר להשלים את קוד המוכנה).

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

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

## תהליך העבודה של האסמבלר

נתאר כעט את אופן העבודה של האסמבלר. בהמשך, יוצג אלגוריתם שלדי למעבר ראשון ושני.

האסמבלר מתחזק שני מערכים, שיקראו להן תМОנות הHorאות (code) ותМОנות הנתונים (data). מערכים אלו נותנים למעשה תМОנה של זיכרון המכונה (כל איבר במערך הוא בגודל מילה של המכונה, ככלומר 24 סיביות). במערך הHorאות בונה האסמבלר את הקידוד של Horאות המכונה שנקראו במהלך המעבר על קובץ המקור. במערך הנתונים מכניס האסמבלר את קידוד הנתונים שנקראו מקובץ המקור (שורות הנחיה מסווג 'data.' ו-'string.') .

האסמבלר משתמש בשני מונחים, שנקראים IC (מונה הHorאות - Instruction-Counter) ו- DC (מונה הנתונים - Data-Counter). מונחים אלו מצביעים על המיקום הבא הפוני במערך הHorאות ובמערך הנתונים, בהתאם. בכל פעע שימושה האסמבלר עובר על קובץ מקור, המונה IC מקבל ערך התחלתי 100, ומונה DC מקבל ערך התחלתי 0. הערך ההתחלתי IC=100 נקבע כדי שקוד המכונה של התוכנית יתאים לטעינה לזכרון (לצורך ריצחה) החל מכתובת 100.

בנוסף, מתחזק האסמבלר טבלה, אשר בה נאשפות כל התוויות בהן נתקל האסמבלר במהלך המעבר על קובץ המקור. לטבלה זו קוראים טבלת-הסמלים (symbol-table). לכל סמל נשמרם בטבלה שם הסמל, ערכו המספריאי, ומאפיינים נוספים (אחד או יותר), כגון המיקום בתMOVות הזיכרון (code או data), וסוג הנראות של הסמל (entry או external).

. במעבר הראשון האסמבלר בונה את טבלת הסמלים ואת השלד של תMOVות הזיכרון (Horאות ונתונים).

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

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

2. שורת Horאה :

האסמבלר מנתח את השורה ומפענח מהי Horאה, ומהן שיטות המיעון של האופרנדים. מספר האופרנדים נקבע בהתאם להוראה שנמצאה. שיטות המיעון נקבעות בהתאם לתחביר של כל אופרנד, כפי שהסביר לעיל במפרט שיטות המיעון. למשל, התו '#' מצין מיעון מיידי, תווית מצינית מיעון ישיר, שם של אוצר מצין מיעון אוצר ישיר, ועוד'.

אם האסמבלר מוצא בשורת Horאה גם הגדרה של תווית, אזי התווית (הסמל) המוגדרת מוכנסת לטבלת הסמלים. ערך הסמל בטבלה הוא IC, והמאפיין הוא code.

כעט האסמבלר קובע לכל אופרנד את ערכו באופן הבא :

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

האסמבלר מכניס למערך Horאות, בכינסה עלייה מצבע מונה Horאות IC, את הקוד הבינאיי המלא של המילה הראשונה של Horאה (בפורמט קידוד כפי שתואר קודם). מילה זו מכילה את קוד הפעולה וה-fun, ואת מספרי שיטות המיעון של אופרנד המקור והיעד. IC מקודם ב-1.

זכור שכאשר יש רק אופרנד אחד (ככלומר אין אופרנד מקור), הסיבות של שיטה המיעון של אופרנד המקור יכילה 0. בדומה, אם זו Horאה ללא אופרנדים (rts, stop), אזי הסיבות של שיטות המיעון של שני האופרנדים יכילה 0.

אם זהה הוראה עם אופרנדים (אחד או שניים), האסטブル "משרינו" מוקם במערך ההוראות עבור מילוט-המידע הנוספת הנדרשת בהוראה זו, ככל שנדירותו, ומקדם את IC בהתחם. כאשר אופרנד הוא בשיטת מיון מיידי או אונר ישיר, האסטブル מוקוד גם את המילה הנוספת המתאימה במערך ההוראות. ואילו בשיטת מיון ישיר או יחס, מילת המידע הנוספת במערך ההוראות נשארת ללא קידוד בשלב זה.

### 3. שורת הנחיה:

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

I. '.data' – האסטブル קורא את רשימת המספרים, המופיעה לאחר '.data', מכניס כל מספר אל מערך הנתונים (בקידוד ביןארי), ומקדם את מצביע הנתונים DC ב-1 עבור כל מספר שהוכנס.

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

II. '.string' – הטיפול ב-'string'. דומה ל-'.data', אלא שקודם ascii של התווים הם אלו המוכנסים אל מערך הנתונים (כל تو במילה נפרדת). לבסוף מוכנס למערך הנתונים הערך 0 (המצין סוף מחזורות). המונה DC מוקדם באורך המחרוזת + 1 (גם התו המסיים את המחרוזת תופס מקום).

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

III. '.entry' – זהה הנחיה לאסטブル לאפיון את התווית הנתונה כאופרנד entry בטבלת הסמלים. בעת הפיקת קבצי הפלט (ראו בהמשך), התווית תירשם בקובץ-h-entries.

لتשומת לב: זה לא נחשכש כשייניה אם בקובץ המקור מופיע יותר מהנחיה entry. אחת עם אותה תווית כאופרנד. המופיעים הנוספים אינם מושיפים דבר, אך גם אינם מפיערים.

IV. '.extern' – זהה הצהרה על סמל (תווית) המוגדר בקובץ מקור אחר, והקובץ הנווכי עושה בו שימוש. האסטブル מכניס את הסמל המופיע כאופרנד לטבלת הסמלים, עם הערך 0 (הערך האמייתי לא ידוע, וייקבע רק בשלב הקישור), ועם המאפיין external. לא ידוע באיזה קובץ נמצאת הגדרת הסמל, ואין זה רלוונטי עבור האסטブル.

لتשומת לב: זה לא נחשכש כשייניה אם בקובץ המקור מופיע יותר מהנחיה מזיהה. אחת עם אותה תווית כאופרנד. המופיעים הנוספים אינם מושיפים דבר, אך גם אינם מפיערים.

لتשומת לב: באופרנד 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 מזיה.).extern

בסוף המעבר הראשון, האסטブル מעדכן בטבלת הסמלים כל סמל המואפיין כ-'data', על ידי הוספה (100) + IC (עשורוני) לערכו של הסמל. הסיבה לכך היא שבתמונה הכלולת של קוד המכונה, תמונה הנתונים מופרdata מתמונה ההוראות, וכל הנתונים נדרשים להופיע בקוד המכונה אחרי כל ההוראות. סמל מסווג data הוא תווית בתמונה הנתונים, והעדכו מוסף לערך הסמל (כלומר כתובתו בזיכרונו) את האורך הכולל של תמונה ההוראות, בתוספת כתובת התחלה הטעינה של הקוד, שהיא 100.

טבלת הסמלים מכילה כעת את ערכי כל הסמלים הנחוצים להשלמת תמונה הזיכרון (למעט ערכים של סמלים חיצוניים).

במעבר השני, האסטブル משלים באמצעות בטבלת הסמלים את קידוד כל המילים במערך ההוראות שטרם קודדו במעבר הראשון. במודל המכונה שלנו אלו הן מילוט-מידע נוספות של ההוראות, אשר מוקודדות אופרנד בשיטת מיון ישיר או יחס.

## אלגוריתם שלדי של האסמבלי

ל釐וד ההבנה של תהליך העבודה של האסמבלי, 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 ההוראות (code), ותמונה  
הנתונים (data). לכל חלק נתזק מונה נפרד: IC (מונה ההוראות) ו- DC (מונה הנתונים).

**בנייה את קוד המכונה כך שיתאים לטיעינה לזכורן החל מכתובת 100.**

בכל מעבר מתחילה לקרוא את קובץ המקור מההתחלתה.

### מעבר ראשון

1. אתחל DC ← 0, IC ← 100.
2. קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עברו ל-17.
3. האם השדה הראשון בשורה הוא תווית? אם לא, עברו ל-5.
4. הדלק דגל "יש הגדרת סמל".
5. האם זהה הנחיה לאחסון נתונים, כלומר, האם הנחית data.או 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 אינו תווית חוקית, או שהסמל כבר נמצא בטבלה, יש להודיע על שגיאה.
7. זהה את סוג הנתונים, קודד אותו בתמונה הנתונים, והגדיל את מונה הנתונים DC על ידי הוספת האורך הכולל של הנתונים שהוגדרו בשורה הנוכחית. חזרו ל-2.
8. האם זו הנחית .extern.או הנחית entry.? אם לא, עברו ל-11.
9. אם זהה הנחית entry. חזרו ל-2 (הנחהה תטופל במעבר השני).
10. אם זו הנחית .extern, הכנס את הסמל המופיע כאופרנד של ההנחהה לתוכן בטבלת הסמלים עם הערך 0, ועם המאפיין external. אם הסמל אינו תווית חוקית, או שהסמל כבר נמצא בטבלה ללא המאפיין, יש להודיע על שגיאה. חזרו ל-2.
11. זהה שורת הוראה. אם יש הגדרת סמל (תוויות), הכנס אותו לטבלת הסמלים עם המאפיין code. ערכו של הסמל יהיה IC (אם הסמל אינו תווית חוקית, או שהסמל כבר נמצא בטבלה, יש להודיע על שגיאה).
12. חפש את שם הפעולה בטבלת שמות הפעולות, ואם לא נמצא, אז הודע על שגיאה בשם ההוראה.
13. נתח את מבנה האופרנדים של ההוראה, וחשב מהו מספר המילים הכולל שתופסת ההוראה בקוד המכונה (נקרא למספר זה L).
14. בנה כעת את הקוד הבינארי של המילה הראשונה של ההוראה, ושל כל מילת- מידע נוספת המקבodata אופרנד במיון מיידי.
15. שמר את הערכיים IC ו- L ייחד עם נתונים קוד המכונה של ההוראה.
16. עדכן L ← IC + L, וחזרו ל-2.
17. קובץ המקור נקרא בשלמותו. אם נמצא שגיאות במעבר הראשון, עצור כאן.
18. שמר את הערכיים הסופיים של IC ושל DC (נקרא להם ICF ו- DCF). השתמש בהם לבניית קבצי הפלט, אחרי המעבר השני.
19. עדכן בטבלת הסמלים את ערכו של כל סמל המופיעים כ- data, ע"י הוספת הערך ICF (ראה הסבר לכך בהמשך).
20. התחל מעבר שני.

### מעבר שני

1. קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עברו ל-7.
2. אם השדה הראשון בשורה הוא סמל (תוויות), דרג עליו.
3. האם זהה הנחית data.או string.? אם כן, חזרו ל-1.
4. האם זהה הנחית entry.? אם לא, עברו ל-6.
5. הוסף בטבלת הסמלים את המאפיין entry למאפייני הסמל המופיע כאופרנד של ההנחהה (אם הסמל לא נמצא בטבלת הסמלים, יש להודיע על שגיאה). חזרו ל-1.

.6. השלם את הקידוד הבינארי של מילוט-המידע של האופרנדים, בהתאם לשיטות המיענו בשימוש. ככל אופרנד בקוד המקור המכיל סמל, מצא את ערכו של הסמל בטבלת הסמלים (אם הסמל לא נמצא בטבלה, יש להודיע על שגיאה). אם הסמל מאופיין *external*, הוסף את **בתובת מילת-המידע הרלוונטי לרשימת מילוט-המידע שמתיחסות לסמל חיצוני**. לפי הורץ, לחישוב הקידוד והכתובות, אפשר להיעזר בערכים IC ו-L של הוראה, כפי ששמרו במעבר הראשוני. חוזר ל-1.

.7. קובץ המקור נקרא בשלמותו. אם נמצא שגיאות במעבר השני, עצור כאן.

.8. בנה את קבצי הפלט (פרטים נוספים בהמשך).

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

```

MAIN:    add    r3, LIST
LOOP:    prn    #48
          lea    STR, r6
          inc    r6
          mov    r3, K
          sub    r1, r4
          bne    END
          cmp    val1, #-6
          bne    %END
          dec    K
          jmp    %LOOP
END:     stop
STR:     .string "abcd"
LIST:    .data   6, -9
          .data   -100
.entry K
K:       .data   31
.extern val1

```

בוצע מעבר ראשון על הקוד לעיל, ובנעה את טבלת הסמלים. כמו כן, נשלים במעבר זה את הקידוד של כל תיונת הנתונים, ושל המילה הראונה של כל הוראה. כמו כן, נקודד מילוט-המידע נוספת של כל הוראה, ככל שקידוד זה אינו תלוי בערך של סמל. את מילוט-המידע שעדיין לא ניתן לקודד במעבר הראשון נסמן ב"??" בדוגמה להלן.

Address (decimal)	Source Code	Explanation	Machine Code (binary)	"A,R,E"
0100	MAIN: add r3, LIST	First word of instruction	001010101101	A
0101		Register r3	000000001000	A
0102		Address of label LIST	?	?
0103	LOOP: prn #48		110100000000	A
0104		Immediate value 48	000000110000	A
0105	lea STR, r6		010000000111	A
0106		Address of label STR	?	?
0107		Register r6	000001000000	A
0108	inc r6		010111000011	A
0109		Register r6	000001000000	A
0110	mov r3, K		000000001101	A
0111		Register r3	000000001000	A
0112		Address of label K	?	?
0113	sub r1, r4		001010111111	A
0114		Register r1	000000000010	A
0115		Register r4	000000010000	A
0116	bne END		100110110001	A
0117		Address of label END	?	?

Address (decimal)	Source Code	Explanation	Machine Code (binary)	“A,R,E”
0118	cmp vall, #-6		000100000100	A
0119		Address of label vall	?	?
0120		Immediate value -6	111111111010	A
0121	bne %END		100110110010	A
0122		Distance to label END	?	A
0123	dec K		010111010001	A
0124		Address of label K	?	?
0125	jmp %LOOP		100110100010	A
0126		Distance to label LOOP	?	A
0127	END: stop		111100000000	A
0128	STR: .string “abcd”	Ascii code ‘a’	000001100001	A
0129		Ascii code ‘b’	000001100010	A
0130		Ascii code ‘c’	000001100011	A
0131		Ascii code ‘d’	000001100100	A
0132		Ascii code ‘\0’	000000000000	A
0133	LIST: .data 6, -9	Integer value 6	0000000000110	A
0134		Integer value -9	111111110111	A
0135	.data -100	Integer value -100	111110011100	A
0136	K: .data 31	Integer value 31	000000011111	A

טבלת הסמלים אחרי מעבר ראשוני היא :

Symbol	Value (decimal)	Attributes
MAIN	100	code
LOOP	103	code
END	127	code
STR	128	data
LIST	133	data
K	136	data
vall	0	external

נבע עתה את המעבר השני. נשלים באמצעות טבלת הסמלים את הקידוד החסר במיללים המסומנים “?”.  
הקידוד הבינארי בצוותו הסופית כאן זהה לקוד שהוצג בתחילת הנושא **“אסמבילר עם שני מעברים”**.

הערה : כאמור, האסמבילר בונה קוד מכונה כך שיתאים לטעינה לזכרון החל מכתובת 100 (עשרוני).  
אם הטעינה בפועל (לצורך הרצת התוכנית) תהיה לכתובה אחרת, יידרשו תיקונים בקוד הבינארי  
בשלב הטעינה, שיוכנסו ביעורת מידע נוסף שהאסמבילר מכין בקבצי הפלט (ראו בהמשך).

Address (decimal)	Source Code	Explanation	Machine Code (binary)	“A,R,E”
0100	MAIN: add r3, LIST	First word of instruction	001010101101	A
0101		Register r3	0000000001000	A
0102		Address of label LIST	000010000101	R
0103	LOOP: prn #48		110100000000	A
0104		Immediate value 48	000000110000	A
0105	lea STR, r6		010000000111	A
0106		Address of label STR	000010000000	R
0107		Register r6	000001000000	A
0108	inc r6		010111000011	A
0109		Register r6	0000010000000	A
0110	mov r3, K		0000000001101	A
0111		Register r3	0000000001000	A
0112		Address of label K	000010001000	R

Address (decimal)	Source Code	Explanation	Machine Code (binary)	“A,R,E”
0113	sub r1, r4		001010111111	A
0114		Register r1	000000000010	A
0115		Register r4	000000010000	A
0116	bne END		100110110001	A
0117		Address of label END	000001111111	R
0118	cmp val1, #-6		000100000100	A
0119		Address of extern label val1	000000000000	E
0120		Immediate value -6	111111111010	A
0121	bne %END		100110110010	A
0122		Distance to label END	000000000101	A
0123	dec K		010111010001	A
0124		Address of label K	000010001000	R
0125	jmp %LOOP		100110100010	A
0126		Distance to label LOOP	111111101001	A
0127	END: stop		111100000000	A
0128	STR: .string “abcd”	Ascii code ‘a’	000001100001	A
0129		Ascii code ‘b’	000001100010	A
0130		Ascii code ‘c’	000001100011	A
0131		Ascii code ‘d’	000001100100	A
0132		Ascii code ‘\0’	000000000000	A
0133	LIST: .data 6, -9	Integer value 6	000000000110	A
0134		Integer value -9	111111110111	A
0135	.data -100	Integer value -100	111110011100	A
0136	K: .data 31	Integer value 31	000000011111	A

טבלת הסמלים אחרי מעבר שני היא :

Symbol	Value (decimal)	Attributes
MAIN	100	code
LOOP	103	code
END	127	code
STR	128	data
LIST	133	data
K	136	data, entry
val1	0	external

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

### קבצי קלט ופלט של האסמבלי

בפעולת של האסמבלר, יש להעביר אליו באמצעות ארגומנטים של שורת הפקודה (command line arguments) רשימה של שמות קבצי מקור (אחד או יותר). אלו הם קבצי טקסט, וביהם תוכניות בתחביר של שפת האסמבלי שהוגדרה בממ"ז זה.

האסמבלר פועל על כל קובץ מקור בנפרד, ויוצר עבورو קבצי פלט כלהלן :

- קובץ object, המכיל את קוד המcona.
- קובץ externals, ובו פרטיים על כל המKENOTOT (הכתובות) בקוד המcona בהם ישAMILT-MIDU SHMKODDUT URZ SEL SMEL SHOHIZHER CHITZONI (SMEL SHOHOFIUS CAOPRND SEL NACHIYT ., EXTERN, OMOPIN).
- בטבלת הסמלים C- (external).
- קובץ entries, ובו פרטיים על כל SMEL SHMOZHAR CNKODOT CNISAH (SMEL SHOHOFIUS CAOPRND SEL NACHIYT ., OMOPIN BETABELT HESMELIM C-).

אם אין בקובץ המקור אפ' הנחיה `extern`, האסמבלר לא יוצר את קובץ הפלט מסוג `externals`.  
אם אין בקובץ המקור אפ' הנחיה `entry`, האסמבלר לא יוצר את קובץ הפלט מסוג `entries`.

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

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

```
assembler x y hello
```

תritz את האסמבלר על הקבצים : `x.y`, `y.x`, `hello.as` :

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

לדוגמא, בהפעלת האסמבלר באמצעות שורת הפקודה : `x.y`, `hello.as` :  
יוצר קובץ פلت `ob.x`, וכן קבצי פلت `ext.y` ו- `ent.x`. ככל שיש הנחיה `entry` או `extern`. בקובץ המקור.  
נציג CUT את הפורמלטים של קבצי הפלט. דוגמאות יובאו בהמשך.

### **פורמט קובץ ה- object**

קובץ זה מכיל את תמונה הזיכרון של קוד המcona, שני חלקים : תМОונת ההוראות ראשונה,  
ואחריה ובצמוד תМОונת הנתונים.

כזכור, האסמבלר מקודד את ההוראות כך שתמונה ההוראות תתאים לטעינה החל מכתובת 100  
(עשורוני) בזיכרון. נשים לב שרך בסוף המ עבר הראשון יודיעים מהו הגולל הכללי של תמונה  
ההוראות. מכיוון שתמונה הנתונים נמצאת אחרי תמונה ההוראות, גודל תמונה ההוראות משפיע  
על הכתובות בתמונה הנתונים. זו הסיבה שבגללה היה צריך לעדכן בטבלת הסמלים, בסוף המ עבר  
הראשון, את ערכי הסמלים המאופינים `c-data` (כזכור, באлогריתם השלדי שהציג לעיל, בצעד 19,  
הוספנו לכל סמל כזה את הערך `ICF`). במעבר השני, בהשלמת הקיזוד של מילות-המידע,  
משתמשים בערכים המעודכנים של הסמלים, המותאמים למבנה המלא והסופי של תמונה הזיכרון.

CUT האסמבלר יכול לכתוב את תמונה הזיכרון בשלמותה לתוכן קובץ פلت (קובץ `h-object`).

השורה הראשונה בקובץ `h-object` היא "cotratt", המכילה שני מספרים (בסיס עשורוני) :  
הראשון הוא האורך הכללי של תמונה ההוראות (במילות זיכרון), והשני הוא האורך הכללי של  
תמונה הנתונים (במילות זיכרון). בין שני המספרים מפריד רווח אחד.  
כזכור, במעבר הראשון, בצעד 18, נשמרו הערכים `ICF` ו- `IDF`. האורך הכללי של תמונה ההוראות  
הוא `100-ICF-IDF`, והאורך הכללי של תמונה הנתונים הוא `IDF`.

השורות הבאות בקובץ מכילות את תמונה הזיכרון. בכל שורה שלושה שדות : כתובות של מילה  
בזיכרון, תוכן המילה, והמאפיין `"A,R,E"`. הכתובת תירשם בסיסי עשורוני באربע ספרות (כולל  
אפסים מובילים). תוכן המילה יירשם בסיסי הקסדצימלי ב-3 ספרות (כולל אפסים מובילים).  
בין השדות בשורה יש רווח אחד.

### **פורמט קובץ ה- entries**

קובץ `h-entries` בניית משורות טקסט, שורה אחת לכל סמל שמאופיין בטבלת הסמלים כ- `entry`.  
בשורה מופיע שם הסמל, ולאחריו ערכו כפי שנקבע בטבלת הסמלים (בסיס עשורוני באربע  
ספרות, כולל אפסים מובילים). בין שני השדות בשורה יש רווח אחד. אין חשיבות לסדר השורות,  
כי כל שורה עומדת בפני עצמה.

### **פורמט קובץ ה- externals**

קובץ `h-externals` בניית משורות טקסט, שורה לכל כתובות בקוד המcona בה יש מילת  
מידע המתיחסת לסמל שמאופיין כ- `external`. כזכור, רשימה של מילות-מידע אלה נבנתה  
במעבר השני (צעד 6 באлогריתם השלדי).

כל שורה בקובץ externals מכילה את שם הסמל החיצוני, ולאחריו הכתובת של מילט-המידע (בבסיס עשרוני באربע ספרות, כולל אפסים מובילים). בין שני השדות בשורה יש רווח אחד. אין חשיבות לסדר השורות, כי כל שורה עומדת בפני עצמה.

لتשומת לב: יתכן ויש מספר כתובות בקוד המכונה בהן מילוט-המידע מתייחסות לאותו סמל חיצוני. לכל כתובות כזו תהיה שורה נפרדת בקובץ externals.

נדגים את הפלטשמייצר האסמבולר עבור קובץ מקור בשם ps.as המתו להלן.

; file ps.as

```
.entry LIST
.extern W
MAIN:    add   r3, LIST
LOOP:     prn   #48
          lea   W, r6
          inc   r6
          mov   r3, K
          sub   r1, r4
          bne   END
          cmp   K, #-6
          bne   %END
          dec   W
.entry MAIN
          jmp   %LOOP
          add   L3, L3
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, בגמר המעבר השני.

Address (decimal)	Source Code	Explanation	Machine Code (binary)	“A,R,E”
0100	MAIN: add r3, LIST	First word of instruction Register r3 Address of label LIST	001010101101 000000001000 000010001000	A A R
0103	LOOP: prn #48		110100000000	A
0104		Immediate value 48	000000110000	A
0105	lea W, r6		010000000111	A
0106		Address of extern label W	000000000000	E
0107		Register r6	000001000000	A
0108	inc r6		010111000011	A
0109		Register r6	000001000000	A
0110	mov r3, K		000000001101	A
0111		Register r3	000000001000	A
0112		Address of label K	000010001011	R
0113	sub r1, r4		001010111111	A
0114		Register r1	000000000010	A
0115		Register r4	000000010000	A
0116	bne END		100110110001	A
0117		Address of label END	000010000010	R
0118	cmp K, #-6		000100000100	A
0119		Address of label K	000010001011	R
0120		Immediate value -6	11111111010	A
0121	bne %END		100110110010	A
0122		Distance to label END	000000001000	A
0123	dec W		010111010001	A
0124		Address of extern label W	000000000000	E
0125	jmp %LOOP		100110100010	A
0126		Distance to label LOOP	111111101001	A
0127	add L3, L3		001010100101	A
0128		Address of extern label L3	000000000000	E
0129		Address of extern label L3	000000000000	E
0130	END: stop		111100000000	A
0131	STR: .string “abcd”	Ascii code ‘a’	000001100001	A
0132		Ascii code ‘b’	000001100010	A
0133		Ascii code ‘c’	000001100011	A
0134		Ascii code ‘d’	000001100100	A
0135		Ascii code ‘\0’	000000000000	A
0136	LIST: .data 6, -9	Integer value 6	000000000110	A
0137		Integer value -9	111111110111	A
0138	.data -100	Integer value -100	111110011100	A
0139	K: .data 31	Integer value 31	000000011111	A

: טבלת הסמלים בגמר המעבר השני היא :

Symbol	Value (decimal)	Attributes
W	0	external
MAIN	100	code, entry
LOOP	103	code
END	130	code
STR	131	data
LIST	136	data, entry
K	139	data
L3	0	external

לחלו תוכן קבצי הפלט של הדוגמה.

הקובץ ob

31 9  
0100 2AD A  
0101 008 A  
0102 088 R  
0103 D00 A  
0104 030 A  
0105 407 A  
0106 000 E  
0107 040 A  
0108 5C3 A  
0109 040 A  
0110 00D A  
0111 008 A  
0112 08B R  
0113 2BF A  
0114 002 A  
0115 010 A  
0116 9B1 A  
0117 082 R  
0118 104 A  
0119 08B R  
0120 FFA A  
0121 9B2 A  
0122 008 A  
0123 5D1 A  
0124 000 E  
0125 9A2 A  
0126 FE9 A  
0127 2A5 A  
0128 000 E  
0129 000 E  
0130 F00 A  
0131 061 A  
0132 062 A  
0133 063 A  
0134 064 A  
0135 000 A  
0136 006 A  
0137 FF7 A  
0138 F9C A  
0139 01F A

הקובץ ps.ent

MAIN 0100  
LIST 0136

הקובץ ps.ext

W 0106  
W 0124  
L3 0128  
L3 0129

## סיכום והנחיות כלליות

- גודל תוכנית המקור הניתנת כקלט לאסטבלר אינו ידוע מראש, ולכן גם גודלו של קוד המוכנה אינו צפוי מראש. אולם כדי להקל בימוש האסטבלר, מוטר להניח גודל מקסימלי. לפיכך יש אפשרות להשתמש במערכות לאחסן תמונות קוד המוכנה בלבך. כל מבנה נתונים אחר (למשל טבלת הסמלים), יש למשה באופן ייעיל וחסכוני (למשל באמצעות רשיימה מקושרת והקצאת זיכרון דינמי).
- השמות של קבצי הפלט צריים להיות תואמים לשם קובץ הקלט, למעט הסיומות. למשל, אם קובץ הקלט הוא asprog אזי קבצי הפלט שיוצרו הם : prog.ob, prog.ext, prog.ent.
- מתכונת הפעלת האסטבלר צריכה להיות כפי הנדרש בממ"ן, ללא שינוי כלשהם. ככלומר, ממשך המשמש יהיה אך ורק באמצעות שורת הפוקודה. בפרט, שמורות קבצי המקור יועברו לתוכנית האסטבלר כארגומנטים (אחד או יותר) בשורת הפוקודה. אין להוסיף תפריטי קלט אינטראקטיביים, חלונות גרפיים למיניהם, ועוד'.
- יש להקפיד לחלק את שימוש האסטבלר למספר מודולים (קבצים בשפט C) לפי משימות. אין לרכז משימות מסוימים במודול אחד. מומלץ לחלק למודולים כגון : מעבר ראשון, מעבר שני, פונקציות עזר (למשל, תרגום לבסיס, ניתוח תחבירי של שורה), טבלת הסמלים, מפת הזיכרון, טבלאות קבועות (קוד הפעולה, שיטות המיעון החוקיקות לכל פעולה, ועוד').
- יש להקפיד ולתעד את השימוש באופן מלא וברור, באמצעות העורות מפורטות בקוד.
- יש לאפשר תווים לבנים עודפים בקובץ הקלט בשפט אסטבלרי. למשל, אם בשורת הוראה יש שני אופרנדים המופרדים בפסיק, אזי לפני ואחרי הפסיק מותר שייהיו רווחים וטאבים בכל כמות. בדומה, גם לפני ואחרי שם הפעולה. מותר גם שורות ריקות. האסטבלר יתעלם מהתווים לבנים מיותרים (כלומר יזלג עליהם).
- הקלט (קוד האסטבלר) עלול להכיל שגיאות תחביריות. על האסטבלר לגלות ולדווח על כל השורות השגויות בקלט. אין לעצור את הטיפול בקובץ קלט לאחר גילוי השגיאה הראשונה. יש להדפיס לפחות הודעות מפורטות לכל הניתן, כדי שאפשר יהיה להבין מה והיכן כל שגיאה. כמובן שאם קובץ קלט מכיל שגיאות, אין טעם להפיק עבورو את קבצי הפלט (ob, ext, ent).

תס ונסלים פרק ההסבירים והגדרת הפרויקט.

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

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

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

**בזה צלח !**