

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 : 2021 א'

קיימות שתי חלופות להגשת מטלות:

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

הסבר מפורט ב"נווה הגשת מטלות מנהה"

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

עליכם ל כתוב תוכנת אסמבילר, עבור שפת אסמבלי 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) עבור מערכת אוביונטו.
3. קובץ makefile. הקימפול חייב להיות עם הקומפיילר gcc והדגלים : -Wall -ansi -pedantic . יש לנפות את כל ההודעות שモוצאת הקומפיילר, כך שהתוכנית תתקמפל ללא כל העוראות או זהירות.
4. דוגמאות הרצה (קלט ופלט)
 - א. קבצי קלט בשפה אסמבלי, וקבצי הפלט שנוצרו מהפעלת האסמבילר על קבצי קלט אלה. יש להציג שימוש במגוון הפעולות וטיפוסי הנתונים של שפת האסמביל.
 - ב. קבצי קלט בשפה אסמבלי המדגימים מגוון רחב של סוגי שגיאות אסמבלי (ולכן לא נוצרים קבצי פלט), ותדריסי המשך המראים את הودעות השגיאה שモוצאת האסמביל.

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

נזכיר מספר היבטים חשובים של כתיבת קוד טוב :

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

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

ומותר 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 רגילה 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 להוות באוטו מחשב עצמו. כיצד "מכיר" המחשב כל כך הרבה שפות? התשובה פשוטה: המחשב מכיר למעשה שפה אחת בלבד: הוראות ונתונים הכתובים בקוד בינארי. קוד זה מאוחסן בגוש בזיכרונו, ונראה כמו רצף של ספרות בינאריות. יחידת העיבוד המרכזית - היע"מ (CPU) - יודעת לפרק את הרצף הזה לקטעים קטנים בעלי משמעות: הוראות, מענים ונתונים.

למעשה, זיכרונו המחשב יכול 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 בזיכרונו שבו נמצאת תוכנית לבין שאר הזיכרונו.

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

היע"מ יודע לבצע קוד שנמצא בפורמט של **שפת מכונה**. זה רצף של ביטים, המהווים קידוד ביןארי של סדרת הוראות המכונה המרכיבות את התוכנית. קוד זה אינו קרייא למשתמש, ולכן לא נוח לקודד (או לקרוא) 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 לכך 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), המתרגם 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 אסambilי יעודית משלו. לפיכך, גם האסambilר (כלי התרגומים) הוא יעודית ושונה לכל יע"מ.

תפקידו של האסambilר הוא לבנות קובץ המכיל קוד מכונה, מקובץ נתון של תוכנית הכתובת בשפת אסambilי. 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 לכתוב אסambilר (כלומר תוכנית המתרגם לשפת מכונה), עבר שפת אסambilי שנגידר כאן במיוחד לצורך הפרויקט.

لتשומת לב : בהסבירים הכלליים על אופן עבודת 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: **עליכם לכתוב את תוכנית האסambilר בלבד. אין לכתוב את תוכניות הקישור והטעינה!!!**

המחשב הדמיוני ושפת האסטREL

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

"חומרה":

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

למעבד 8 אוגרים כלליים, בשמות : r0, r1, r2, r3, r4, r5, r6, r7. הסיבית ה-12 שאל אוגר הוא סיביות. הסיבית ה-0 של אוגר ה-12 שאל אוגר ה-0. השמות האוגרים כתובים תמיד עם אות 'r' קטנה. המשמעותית ביותר במס' 11. 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 תמיכה במספרים ממשיים. האריתמטיקה נעשית בשיטת המשלים ל-2 (2's complement). כמו כן יש תמיכה 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 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	funct				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)**, ו**פונקציה (funct)**.

להלן טבלת הפעולות :

שם הפעולה	funcf (בסיס עשרוני)	opcode (בסיס 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 המכוна 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 שմבדיל בינהן 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 להן אותן קוד-פעולה. אם קוד-הפעולה משמש ל פעולה 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 יינתן 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 המכוна של הוראה, בנוסף למילה הראשונה. לכל אופרנד של הוראה נדרש מילוט-מידע **אחד** נוספת **נוספת**. כאשר בהוראה יש שני אופרנדים, קודם תופיע מילוט-המידע של אופרנד המקור, ולאחר מכן מילוט-המידע של אופרנד היעד. להלן המפרט של שיטות המיעון.

מספר	שיטת המיעון	תוכן מילת-המידע הנוספת	תחביר האופרנד באסמלבי	דוגמה
0	מיעון מיידי (immediate)	밀ת-מידע נוספת נוספת של ההוראה מכילה את האופרנד עצמו, שהוא מספר שלם בסיס המשלים ל-2, ברוחב של 12 סיביות	האופרנד מתחיל ב'תו #' ולאחריו ובצמוד אליו מופיע מספר שלם בסיס עשרוני.	mov #1, r2 בוגמה זו האופרנד הראשון של ההוראה (אופרנד המקורי) נתון בשיטת מיון מיידי. ההוראה כתובת את הערך 1 אל אוגר r2.
1	מיעון ישיר (direct)	밀ת-מידע נוספת נוספת של ההוראה מכילה כתובות בזיכרון. המילה כתובת זו בזיכרון היא האופרנד. הכתובת מיוצגת כמספר <u>לא סימן</u> ברוחב של 12 סיביות.	האופרנד הוא <u>תוויות</u> שכבר הוגדרה, או שתוגדר בהמשך הקובץ. ההגדרה נעשית על ידי כתיבת התווית בתחילת השורה של הוראה, או בתחילת השורה '.string', או בתחילת השורה 'EXTERN', או באמצעות אופרנד של הנחית 'EXTERN'. התווית מייצגת באופן סימבולי כתובות בזיכרון.	הורה הבאה מדירה את התווית x : x: .data 23 ההוראה : dec x מקטינה ב-1 את תוכן המילה שבסכמתו של x בזיכרון ("משתנה" x). הכתובת x מקודדת במייל-המידע הנוספת. <u>דוגמה נוספת :</u> ההוראה jmp next מבצעת קפיצה אל השורה בה מוגדרת התווית next (כלומר ההוראה הבאה שתבצעה נמצאת בכתובת next). הכתובת next מקודדת במייל-המידע הנוספת.
2	מיעון יחסית (relative)	שיטה זו רלוונטית אך ורק להוראות המבצעות קפיצה (הסתעפות) להוראה אחרת. מדובר בהוראות עם קוד-פעולה 9 בלבד : jmp, bne, jsr. <u>לא ניתן להשתמש בשיטה זו בהוראות עם קוד-פעולה אחרים.</u> בשיטת זו, יש בקידוד ההוראה מילת-מידע נוספת, המכילה את מרחק הקפיצה, במילוט זיכרון, מamilat-hmidu-nocchiyah al milah hareshona shel horahah habekoshet (ההוראה הבאה לביצוע).	האופרנד מתחיל ב'תו %' ולאחריו ובצמוד אליו מופיע שם של תווית. התווית מייצגת באופן סימבולי כתובות של הוראה <u>בקובץ המקור הנוכחי של התוכנית</u> .	דוגמה זו, ההוראה jmp %next בוגמה זו, ההוראה jmp מבצעת קפיצה אל השורה בה מוגדרת התווית next (כלומר ההוראה הבאה שתבצעה נמצאת בכתובת next). נניח, לדוגמה, כי ההוראה jmp שבוגמה נמצאת בכתובת 500 (עשרהוני). כמו כן, נניח כי התווית next מוגדרת בקובץ המקור הנוכחי בכתובת 300 (עשרהוני). מרחק הקפיצה ממילת-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	דוגמאות
3	מייעון אוגר ישיר (register direct)	האופרנד הוא אוגר. AMILIT-MIDU NOSFAT SHL HORA MCILAH BSIYIOT 7-0 BIT DOLK ICHID HEMIYCIG AT HOGER HMTAIM. SIYIOT 0 TDLOK AMMDOR BAOGER 0Z, SIYIOT 1 TDLOK AMMDOR BAOGER 1Z VOKI. SIYIOT 11-8 YHIO TMID MAOFSEOT	האופרנד הוא שם של אוגר. בדוגמה זו, ההוראה מפסיק את האוגר r1. המילה הנוספת של ההוראה תכיל (בבינארית) 000000000010 <u>דוגמה נוספת:</u> mov #1, r2 האופרנד השני של ההוראה (אופרנד היעד) נתון בשיטת מייעון אוגר ישיר. ההוראה כתובת ישירות. הערך המיידי 1- אל אוגר r2. המילה הנוספת השנייה של ההוראה תכיל (בבינארית) 00000000100	clr r1 clr 0Z, HORA MAFSAT AT HOGER r1. HAMILAH HNOSFAT SHL HORA HORA TAAL (BINARIA) mov #1, r2

מפורט הוראות המכונה:

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

הוראות המכונה מתחלקות לשולש קבוצות, לפי מספר האופרנדים הנדרשים לפעולה.

קבוצת ההוראות הראשונה:

אליהן הוראות המקבלות שני אופרנדים.

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

הוֹרָאָה	opcode	funct	הפעולה המתבצעת	דוגמאות	הסביר 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 שבכנתובת בזיכרון) אל אוגר r1.
cmp	1		מבצע השוואת בין שני האופרנדים. ערך אופרנד היעד (השני) מופחת מערך אופרנד המקור (הראשון), ללא שמירת תוצאת החישור. פעולה החישור 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 הנקחי של 0Z.
sub	2	11	אופרנד היעד (השני) מקבל את תוצאה החישור של אופרנד המקור (הראשון) מאופרנד היעד (השני).	sub #3, r1	אוגר 0Z מקבל את תוצאה החישור של הקבוע 3 מתוכנו הנקחי של האוגר r1.
lea	4		lea הוא קייזר (ראשי תיבות) של load effective address. פעולה זו מציבה את מען הזיכרון המיוצג על ידי התווית שבאופרנד הראשון (המקור), אל האופרנד השני (היעד).	lea HELLO, r1	המען שמייצגת התווית HELLO מוצב לאוגר r1.

קבוצת ההוראות השניה:

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

הוֹרָאָה	opcode	funct	הפעולה המתבצעת	דוגמָה	הסביר הדוגמה
clr	5	10	איפוס תוכן האופרנד.	clr r2	האגר 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 באגר 2 ז' מתחפה.
inc	5	12	הגדלת תוכן האופרנד באחד.	inc r2	תוכן האגר 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 ההוראה שנמצאת בمعنى המיצג על ידי האופרנד. כלומר, כתוצאה מביצוע ההוראה, מצביע התוכנית (PC) מקבל את כתובות יעד הקפיצה.	jmp %Line	PC ← PC+distanceTo(Line) בשיטת מעון יחסי, המרחק לתוכית Line מהתו סוף למצויע התוכנית ולפיכך ההוראה הבאה שתתבצע תהיה בمعنى Line.
bne	9	11	אם ערך הדגל Z באוגר הסטטוס (PSW) הוא 0, אז PC ← address(Line) מצביע התוכנית קיבל את כתובות התווית Line, ולפיכך ההוראה הבאה שתתבצע תהיה בمعنى Line.	bne Line	אם ערך הדגל Z באוגר הסטטוס (PSW) הוא 0, אז PC ← address(Line) מצביע התוכנית קיבל את כתובות התווית Line, ולפיכך ההוראה הבאה שתתבצע תהיה בمعنى Line.
jsr	9	12	קריאה לשגרה (סבוריינית). כתובות ההוראה שאחרי הוראות jsr הנוכחות (PC+2) נדחפת לתוך המחסנית שבירכון המחשב, ומצויע התוכנית (PC) מקבל את כתובות השגרה. <u>הערה:</u> חוזרת מהשגרה מתבצעת באמצעות הוראות rts, תוך שימוש בכתובות שבמחסנית.	jsr SUBR	push(PC+2) PC ← address(SUBR) מצביע התוכנית קיבל את כתובות התווית SUBR, SUBR, וההוראה הבאה שתתבצע תהיה בمعنى SUBR. SUBR נשמרת במחסנית.
red	12		קריאה שלתו מהקלט הסטנדרטי (stdin) אל האופרנד.	red r1	קוד ה-ascii של התו הנקרא r1 מהקלט ייכנס לאוגר r1.
prn	13		הדפסת התו הנמצא באופרנד, אל הפלט הסטנדרטי (stdout).	prn r1	יודפס לפט התו (קוד ascii)r1 הנמצא באוגר r1

קבוצת ההוראות השלישייה:

אלו הן ההוראות 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 השויות לקבוצה זו הן: rts, stop

הוֹרָאָה	opcode	הפעולה המתבצעת	דוגמָה	הסביר הדוגמה
rts	14	מתבצעת חזרה משגרה. הערך שבראש המחסנית של המחשב מזוכה מן המחסנית, ומוכנס למצביע התוכנית (PC). <u>הערה:</u> ערך זה נכנס למחסנית בקריאה לשגרה ע"י הוראת jsr.	rts	PC ← pop() ההוראה הבאה שתתבצע תהיה זו שאחרי הוראת jsr שקרה לשגרה. התוכנית עוצרת מיידית.

מבנה תכנית במשפט אסטנבי :

תכנית במשפט אסטנבי בנויה ממשפטים (statements). קובץ מקור במשפט אסטנבי מורכב משורות המכילות משפטי של השפה, אשר כל המשפט מופיע בשורה נפרדת. כלומר, הפרדה בין המשפט למשפט בקובץ המקור הינה באמצעות התו 'ז' (שורה חדשה).

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

יש ארבעה סוגי משפטיים (שורות בקובץ המקור) במשפט אסטנבי, והם :

סוג המשפט	הסבר כללי
משפט ריק	זהו שורה המכילה אך ורק תוויים לבנים (whitespace), ככלומר רק את התווים ' ' ו- 't' (רווחים וטאים). יתכן ובסורה אין אףתו (למעט התו ז'), ככלומר השורה ריקה.
משפט הערת	זהו שורה בה התו הראשון הינו ';' (נקודה פסיק). על האסטנבל להתעלם לחולטין משורה זו.
משפט הקיימת	זהו משפט המנחה את האסטנבל מה עליו לעשות כשהוא פועל על תוכנית המקור. יש מספר סוגים של משפטי הקיימת. משפט הקיימת עשוי לגורם להקצת זיכרונו ואתחול משתנים של התוכנית, אך הוא אינו מייצר קידוד של הוראות מכונה המיועדות לביצוע בעת ריצת התוכנית.
משפט הוראה	זהו משפט המ מייצר קידוד של הוראות מכונה לביצוע בעת ריצת התוכנית. המשפט מורכב ממשם ההוראה (פעולה) שעל המעבד לבצע, והאופרנדים של ההוראה.

כעת נפרט יותר לגבי סוגי המשפטיים השונים.

משפט הקיימת:

משפט הקיימת 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 הקיימת. לאחר שם הקיימת יופיעו פרמטרים (מספר הפרמטרים בהתאם להקיימת). שם של הקיימת מתחילה בתו '.' (נקודה) ולאחריו תוים באותיות קטנות (lower case) בלבד.

יש ארבעה סוגיים (שמות) של משפטי הקיימת, והם :

1. ההקיימת 'data'.

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

.data 7, -57, +17, 9

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

המשפט 'data' מנהה את האסטנבל להקצות מקומות בתמונות הנתונים (data image), אשר בו יוחסנו הערכים של הפרמטרים, ולאחר מכן מושכים בהתאם הנתונים, בהתאם למספר הערכים. אם בהנחתה data. מוגדרת תווית, אז תווית זו מקבלת את ערך מונה הנתונים (לפניהם הקידום),

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

כלומר אם נכתב :

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

אז יוקצו בתמונה הנתונים ארבע מילימ רצופות שיכילו את המספרים שמופיעים בהנחיה. התויה XYZ מזוהה עם כתובות המילה הראשונה.

אם נכתב בתוכנית את הוראה :

mov XYZ, r1

אז בזמן ריצת התוכנית יוכנס לאוגר 1 ז' ערך 7.

ואילו הוראה :

lea XYZ, r1

תכנס לאוגר 1 ז' את ערך התויה XYZ (כלומר הכתובת בזיכרון בה מאוחסן הערך 7).

2. ההנחיה 'string'

להנחיה 'string', פרמטר אחד, שהוא מחוזות חוקית. תווים המחרוזת מקודדים לפי ערכי-ascii המתאים, ומוכנסים אל תמונה הנתונים לפי סדרם, כל تو במילה נפרדת. בסוף המחרוזת יתווסף התן '0' (ערך המספרי 0), המסמן את סוף המחרוזת. מונה הנתונים של האסמלר יקודם בהתאם לערך המחרוזת (בתוספת מקום אחד עבור התו המסיים). אם בשורת ה命令oga מוגדרת תויה, אז תויה זו מקבלת את ערך מונה הנתונים (לפניהם הקידום) ומוכנסת אל טבלת הסמלים, בדומה כפי שנעשה עבור 'data'. (כלומר ערך התויה יהיה הכתובת בזיכרון שבה מתילה המחרוזת).

לדוגמה, ההנחיה :

STR: .string "abcdef"

מקצת בתמונה הנתונים רצף של 7 מילימ, ומאתחלת את המילימ לקוד ascii של התווים לפי הסדר במחוזות, ולאחריהם ערך 0 לסימון סוף מחוזות. התויה STR מזוהה עם כתובות התחלה המחרוזת.

3. ההנחיה 'entry'

להנחיה 'entry', פרמטר אחד, והוא שם של תויה המוגדרת בקובץ המקור הנוכחי (כלומר תויה שמקבלת את ערכיה בקובץ זה). מטרת ההנחיה entry. היא לאפיין את התויה זו באופן שיאפשר לקוד אסמלרי הנמצא בקובץ מקור אחרים להשתמש בה (כאופרנד של הוראה).

לדוגמה, השורות :

HELLO: .entry HELLO
 add #1, r1

מודיעות לאסמלר שאפשר להתייחס בקובץ אחר לתויה HELLO המוגדרת בקובץ הנוכחי.

בתשומת לב : תויה המוגדרת בתחילת שורת entry. הינה חסרת משמעות והאסמלר מתעלם מהתויה זו (אפשר שאסמלר יוציא הודעה אחרת).

4. ההנחיה 'extern'

להנחיה 'extern', פרמטר אחד, והוא שם של תויה שאינו מוגדרת בקובץ המקור הנוכחי. מטרת ההוראה היא להודיע לאסמלר כי התויה מוגדרת בקובץ מקור אחר, וכי קוד האסמלר בקובץ הנוכחי עושה בתויה שימוש.

נשים לב כי הינה זו תואמת להנחיה 'entry.' המופיעה בקובץ בו מוגדרת התווית. בשלב הקישור תבוצע התאמה בין ערך התווית, כפי שנקבע בקוד המכונה של הקובץ שהגדיר את התווית, לבין קידוד ההוראות המשמשות בתווית בקבצים אחרים (שלב הקישור אינו רלוונטי לም"ן זה).

לדוגמא, משפט ההנחיה 'extern.' התואם לשפט ההנחיה 'entry.' מהדוגמה הקודמת יהיה:

.extern HELLO

הערה: לא ניתן להגדיר באותו הקובץ את אותה התווית גם כ-'entry' וגם כ-'extern' (בדוגמאות לעיל, התווית HELLO).

תשומת לב: תווית המוגדרת בתחילת שורת 'extern'. הינה חסרת משמעות והאסמבלר **מתעלם** מהתווית זו (אפשר שהאסמבלר יוציא הודעה אחרת).

משפט הוראה:

משפט הוראה מורכב מחלקים הבאים:

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

אם מוגדרת תווית בשורת הוראה, אז היא תוכנס אל טבלת הסמלים. ערך התווית יהיה מען המילה הראשונה של הוראה 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 (אחד או יותר).

כאשר יש שני אופרנדים, האופרנדים מופרדים זה מזו בטו ' ' (פסיק). בדומה להנחיה 'data.', לא **חייבת להיות הצמדה של האופרנדים לפסיק**. כל כמות של רווחים ו/או ט-abs משני 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 האסמבלר

תוויות:

תווית היא סמל שמוגדר בתחילת משפט הוראה' או בתחילת הנחיה 'data.' או 'string.'.
תווית חוקית מתחילה באות אלפביתית (גדולה או קטנה), ולאחריה סדרה כלשהי של אותיות אלפביתיות (גדולות או קטנות) ו/או ספרות. האורך המקסימלי של תווית הוא 31 תווים.

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

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

לדוגמא, התוויות המוגדרות להלן הן תוויות חוקיות.

hEllo:

x:

He78902:

لتשומת לב : מילים שמורות של שפט האסמלבי (כלומר שם של פעולה או החלטה, 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 מודול .extern. כלשהו בקובץ הנוכחי).

מספר :

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

מחuzeות :

מחuzeות חוקית היא סדרת תווים ascii נראים (שניינטם להדפסה), המוקפים במרכאות כפولات (המרכאות אינן נחובות חלק מהמחuzeות). דוגמה למחuzeות חוקית : “hello world”.

סימנו המילים בקוד המכונה באמצעות המאפיין ”A,R,E”

האסמלבר 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 הנוכחי אינו מתאים ויש צורך לתקן אותו. לדוגמה, מילת-המידע של אופrnd בשיטות מעון ישיר לא תהיה נכון, כי הctaובת השתנתה.

הຽון 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 האסמלבי. תיקונים כאלה נעשים בשלב הקישור והטעינה של הקוד (אנו לא מטפלים בכך בממ"ז זה), אולם על האסמלבר להוטף מידע בקוד המכונה שיאפשר לזהות את הנកודות בקוד בהן נדרש תיקון.

לצד כל מילה בקוד המכונה, האסמלבר 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 יייטה 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 יייטה בפועל קוד המכונה של התוכנית בעת ביצועה (למשל, מילת-מידע המכילה כוותבת 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 תואפיין על ידי האות R או E (תלו依 אם האופrnd בקוד האסמבלי הוא תווית מקומית או סמל חיצוני).

אסמבלר עם שני מעברים

כאשר מקבל האסמבלר כקלט תוכנית בשפת אסמבלי, עליו לעבור על התוכנית פעמיים. במעבר הראשון, יש לזהות את הסמלים (תוויות) המופיעים בתוכנית, ולתת לכל סמל ערך מספרי שהוא המען בזיכרנו שהסמל מייצג. במעבר השני, באמצעות ערכי הסמלים, וכן קוד-הפעולה ומספריו האוגרים, בונים את קוד המכונה.

לדוגמה : האסמבלר 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   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

```

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

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		000000001101	A
0111		Register r3	000000001000	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 val1, #-6		000100000100	A
0119		Address of extern label val1	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	111111110111	A
0135	.data -100	Integer -100	111110011100	A
0136	K: .data 31	Integer 31	000000011111	A

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

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

למשל, בקוד לעיל, האסמבילר אינו יכול לדעת שהסמל END אמרור להיות משוייך למספר 127 (עשרוני), ושהסמל K אמרור להיות משוייך למספר 136, אלא רק לאחר שנקראו כל שורות התוכנית.

לכן מפרידים את הטיפול של האסמבילר 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, בערךיהם 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 מספרי, שהוא מען בזיכרון.

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

עבור 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
vall	0	external

لتשומת לב: תפקיד האסמבילר, על שני המבערים שלו, לתרגם קובץ מקור לקוד בשפת מכונה. בגמר פעולות האסמבילר, התוכנית טרם מוכנה לטעינה ל זיכרון לצורך ביצוע. קוד המכונה חייב לעמוד לשלבי הקישור/טיענה, ורק לאחר מכן לשלב הביצוע (שלבים אלה אינם חלק מהמשמעות).

המעבר הראשון

במעבר הראשון נדרשים כלליים כדי לקבוע איזה מען ישוק לכל סמל. העיקרונות הבסיסיים הוא לספור את המיקומות בזיכרון, אותן תופעות ההוראות. אם כל הוראה תיטען בזיכרון למקום העוקב להוראה הקודמת, תציג ספריה כזאת את מען ההוראה הבאה. הספריה נעשית על ידי האסטבלר ומוחזקת במונה ההוראות (IC). ערכו ההתחלתי של IC הוא 100 (עשרה), ולכן קוד המכמה של ההוראה הראשונה נבנה כך שייטען לזכרון החל ממען 100. IC מעדכן בכל שורת ההוראה המקצת מקום בזיכרון. לאחר שהאסטבלר קובע מהו אורך ההוראה, 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 שימושיות שונות, בכל אחת משיטות המיון, וכן יתאים לה קידודים שונים לפי שיטות המיון. לדוגמה, פעולה ההזזה `STW` יכולה להתייחס להעתקת תוכן תא זיכרון לאוגר, או להעתקת תוכן אוגר לאוגר אחר, וכן הלאה. ככל אפשרות כזו של `STW` עשוי 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 השדות ביחד מילוה אחת או יותר בקוד המכונה.

כאשר נתקל האסטבלר בתוויות המופיעות בתחילת ההוראה, הוא יודע שלפנוי הגדרה של תווית, ואז הוא משייך לה מען – תוכנו הנוכחי של IC. כך מקבלות כל התוויות את מענייהם בעת ההגדירה. 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 מטבלת הסמלים.

הוראה יכולה להתייחס גם לסמל שטרם הוגדר עד כה בתוכנית, אלא יוגדר רק בהמשך התוכנית. להלן, לדוגמה, הוראות 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-המیدע נוספת מיידי, או אוגר, וכן את הקידוד הבינאי של כל הנתונים (המתפללים מההנחיות `,string`, `,data`).

המעבר השני

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

לשם כך מבצע האסטבלר מעבר נוסף (מעבר שני) על כל קבוע המקור, וمعدכן את קוד המכמה של האופרנדים המשתמשים בסמלים, באמצעות ערכי הסמלים מטבלת הסמלים. בסוף המעבר השני, תהיה התוכנית מותרגמת בשלמותה לקוד מכונה.

הפרדת הוראות ונתונים

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

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

האסםבלר שלנו חייב להפריד, בקוד המוכנה שהוא מיצר, 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 הפרדה.

גילוי 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 מודוא האסםבלר שככל סמל מוגדר פעם אחת בדיק.

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 בנוסחה "יוטר 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).

لتשומתך: האסםבלר אינו עובד את פעולתו אחרי שנמצאה השגיאה הראשונה, אלא ממשיך לעבור על הקלט כדי לגנות שגיאות נוספות, ככל שישן. כמובן שאין כל טעם לייצר את קבצי הפלט אם נתגלו שגיאות (ממילא 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 מיעון חוקיות עבור אופרנד היעד	שיטות מיעון 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) ותMOVות הנתונים (data). מערכים אלו נותנים למשה תMOונה של זיכרון המכונה (כל איבר במערך הוא בגודל מילה של המכונה, ככלומר 24 סיביות). במערך הHorאות בונה האסמבלר את הקידוד של הHorאות המכונה שנקראו במהלך המעבר על קובץ המקור. במערך הנתונים מכניס האסמבלר את קידוד הנתונים שנקראו מקובץ המקור (שורות הנחיה מסוג 'data.' ו-'string.').

האסמבלר משתמש בשני מונחים, שנקראים IC (МОונה הHorאות - Instruction-Counter) ו- DC (МОונה הנתונים - Data-Counter). מונחים אלו מצבעים על המיקום הבא הפוני במערך הHorאות ובמערך הנתונים, בהתאם. בכל פעם כשמהלך האסמבלר עבר על קובץ מקור, המונה IC מקבל ערך התחלתי 100, והМОונה 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 המספרי, ומאפיינים נוספים (אחד או יותר), כגון המיקום בתMOVה 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 שורה הערה : האסמבלר מתעלם מהשורה וועבר לשורה הבאה.

2. שורת Horאה :

האסמבלר מנתח את השורה ומפענח מהי Horאה, ומהן שיטות המידע של האופרנדים. מספר האופרנדים נקבע בהתאם להוראה שנמצאה. שיטות המידע נקבעות בהתאם לתחביר של כל אופרנד, כפי שהוסבר לעיל במפרט שיטות המידע. למשל, התו '#' מצין מידע, תווית מצינית מידע ישיר, שם של אוצר מצין מידע 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 (מידע ישיר) – האופרנד הוא ערך התווית כפי שמופיע בטבלת הסמלים (ייתכן והסמל טרם נמצא בטבלת הסמלים, במידה והוא יוגדר רק בהמשך התוכנית).
- אם זה התו '#' ואחריו מספר (מידע מיד) – האופרנד הוא המספר עצמו.
- אם זו שיטה מידע אחרת – ערכו של האופרנד נקבע לפי המפרט של שיטה המידע (ראו תאור שיטות המידע לעיל)

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

זכור שכאשר יש רק אופרנד אחד (ככלומר אין אופרנד מקור), הסיבות של שיטת המידע של אופרנד המקור יכילה 0. בדומה, אם זה Horאה ללא אופרנדים (stop, ts, etc.), אזי הסיבות של שיטות המידע של שני האופרנדים יכילה 0.

אם זהה הוראה עם אופרנדים (אחד או שניים), האסטבלר "משרינו" מוקם במערך ההוראות עבור מילוט-המידע הנוספת הנדרשות בהוראה זו, ככל שנדרשות, ומקדם את IC בהתחם. כאשר אופרנד הוא בשיטת מיון מיידי או אוניג'ישיר, האסטבלר מוקדם גם את המילה הנוספת המתאימה במערך ההוראות. ואילו בשיטת מיון ישר או יחס, מילת המידע הנוספת במערך ההוראות נשארת ללא קידוד בשלב זה.

3. שורת הנחיה:

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

- I. '.data'.
האסטבלר קורא את רשימת המספרים, המופיעה לאחר '.data.', מכניס כל מספר אל מערך הנתונים (בקידוד ביןארי), ומקדם את מצביע הנתונים DC ב-1 עבר כל מספר שהוכנס.
אם בשורה '.data.' מוגדרת גם תווית, אז 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 (כלתו במילה נפרדת). לבסוף מוכנס למערך הנתונים הערך 0 (המצין סוף מחוזות). המונה DC מוקדם באורך המחרוזות + 1 (גם התו המסיים את המחרוזת תופס מקום).
- III. '.entry'.
זהי הנחיה לאסטבלר לאפיין את התווית הנתונה כאופרנד כ-entry בטבלת הסמלים. בעת הפיקת קבוע הפלט (ראו בהמשך), התווית תירשם בקובץ ה-entries.
لتשומת לב: זה לא נחשב לשגיאה אם בקובץ המקור מופיעה יותר מהנחיה entry. אחת עם אותה תווית כאופרנד. המופיעים הנוספים אינם מושיפים דבר, אך גם אינם מפריעים.
- IV. '.extern'.
זהי הצהרה על סמל (תווית) המוגדר בקובץ מקור אחר, והקובץ הנוכחי עושה בו שימוש. האסטבלר מכניס את הסמל המופיע כאופרנד לטבלת הסמלים, עם הערך 0 (הערך האמייתי לא ידוע, וייקבע רק בשלב הקישור), ועם המאפיין external. לא ידוע באיזה קבוע נמצאת הגדרת הסמל, ואין זה רלוונטי עבור האסטבלר.
لتשומת לב: זה לא נחשב לשגיאה אם בקובץ המקור מופיעה יותר מהנחיה extern. אחת עם אותה תווית כאופרנד. המופיעים הנוספים אינם מושיפים דבר, אך גם אינם מפריעים.
لتשומת לב: באופרנד של הוראה או של הנחיה entry, מותר להשתמש בסמל אשר יוגדר בהמשך הקובץ (אם באופן ישר על ידי הגדרת תווית, ואם באופן עקיף על ידי הנחיה וextern.).

בסוף המעבר הראשון, האסטבלר מעדכן בטבלת הסמלים כל סמל המופיע כ-.data, על ידי הוספה (100) + IC (עשרות) לערכו של הסמל. הסיבה לכך היא שבתמונה הכלולת של קוד המכונה, תמונות הנתונים מופרdatת 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, שהוא 100.

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

במעבר השני, האסטבלר משלים באמצעות טבלת 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 ישר או יחס.

אלגוריתם שלדי של האסמבלי

ל釐וד ההבנה של תהליך העבודה של האסמבלי, נציג להלן אלגוריתם שלדי למעבר הראשוני ולמעבר השני.

لتשומת לב: אין חובה להשתמש דווקא באלגוריתם זה.

כאמור, אנו מחלקים את תמונה קוד המcona לשני חלקים: **תמונה ההוראות (code)**, ו**תמונה הנתונים (data)**. לכל חלק נתזקק מונה נפרד: IC (מונה ההוראות) ו- DC (מונה הנתונים).

בנייה את קוד המcona כך שיתאים לטיעינה לזכור החל מכתובת 100.

בכל מעבר מתחילה לקרוא את קובץ המקור מההתחלתה.

מעבר ראשוני

1. $DC \leftarrow 0, IC \leftarrow 100$.
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, למשל, האם הונית `data`. או `string`? אם לא, עברו ל-8.
6. אם יש הגדרת סמל (תווית), 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, קודו אוטם בתמונה הנתונים, והגדיל את מונה הנתונים DC על ידי הוספת האורך הכולול הנתונים שהוגדרו בשורה הנוכחית. חוזר ל-2.
8. האם זו הנחיה זואנטית `extern`. או הנחיה זואנטית `entry`? אם לא, עברו ל-11.
9. אם זהה הנחיה זואנטית `entry`. חוזר ל-2 (הנחהיה תטופל במעבר השני).
10. אם זו הנחיה זואנטית `extern`, הכנס את הסמל המופיע כאופרנד של ההנחה לתוכן טבלת הסמלים עם הערך 0, ועם המאפיין `external`. אם הסמל אינו תווית חוקית, או שהסמל כבר נמצא בטבלה ללא המאפיין `external`, יש להודיע על שגיאה. חוזר ל-2.
11. זהה שורת הוראה. אם יש הגדרת סמל (תווית), 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 הזודע על שגיאה בשם ההוראה.
13. נתח את מבנה האופרנדים של ההוראה, וחשב מהו מספר המילים הכוללת שתופותה ההוראה בקוד המcona (נקרא למספר זה L).
14. בנה כעת את הקוד הבינארי של המילה הראשונה של ההוראה, ושל כל מילת- מידע נוספת המקבצת אופרנד במיון מיידי.
15. שומר את הערכיהם IC ו- L יחד עם נתונים קוד המcona של ההוראה.
16. עדכן $L \leftarrow IC + L$, וחזור ל-2.
17. קובץ המקור נקרא בלטומו. אם נמצא שגיאות במעבר הראשוני, עוזר כאן.
18. שומר את הערכים 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 המופיע כ- `data`, או `string`. 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 (תווית), דרג עליו.
3. האם זהה הנחיה `data`. או `string`? או `extern`? אם כן, חוזר ל-1.
4. האם זהה הנחיה `entry`? אם לא, עברו ל-6.
5. הוסף בטבלת הסמלים את המאפיין entry למאפייני הסמל המופיע כאופרנד של ההנחה (אם הסמל לא נמצא בטבלת הסמלים, יש להודיע על שגיאה). חוזר 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 כתובת מילת-המידע הרלוונטייה לרשימת מילוט-מידע שמתייחסות לSymbol חיצוני. לפי הורץ, לחישוב הקידוד והכתובות, אפשר להיעזר בערכיהם IC ו-L של הוראה, כפי ששמרו במעבר הראשוני. חוזר ל-1.

7. קובץ המקור נקרא בשמותיו. אם נמצא שגיאות במעבר השני, עוזר 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, ובניה את טבלת הסמלים. כמו כן, נשלים במעבר זה את הקידוד של כל תמונות הנתונים, ושל המילה הראונה של כל הוראה. כמו כן, נקודד מילוט-מידע נוספת של כל הוראה, ככל קידוד 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 val1, #-6		000100000100	A
0119		Address of label val1	?	?
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	000000000010	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 ראשון היא :

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

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

הערה : כאמור, האסמבלר בונה קוד מכונה כך שייתאים לטיעינה לזכרוון החל מכתובת 100 (עשרה).
אם הטיעינה בפועל (לצורך הרצת התוכנית) תהיה לכתובות אחרות, יידרשו תיקוניים בקוד הבינארי
בשלב הטיעינה, שיוכנסו בעזרת מידע נוסף שהאסמבller 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	000001000000	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	11111111010	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 בມמ"ז זה.

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

- קובץ object, המכיל את קוד המcona.
- קובץ externals, ובו פרטים על כל המKENOTOT (הכתובות) בקוד המcona בהם ישAMILT-MIDU SHMKODDUT URK SEL SMEL SHOZOCHER CHITZONI (SMEL SHOOFIUS CAOPRND SHL HNCHIYT .,extern ., וMAOFIIN BETBALT HSMELIM C-).
- קובץ entries, ובו פרטים על כל SMEL SHMOZCHER CNKODOT CNISAH (SMEL SHOOFIUS CAOPRND SHL HNCHIYT .,entry ., וMAOFIIN BETBALT HSMELIM C-).

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

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

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

```
assembler x y hello
```

הרץ את האסמבלר על הקבצים : `x.as`, `y.as`, `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-`entries`, והסיומת
`"ext"`. עבור קובץ-`externals`.

לדוגמה, בהפעלת האסמבלר באמצעות שורת הפקודה : `x` `assembler` `x` `y` `hello`.
يוצר קובץ פلت `ob.x`, וכן קבצי פلت `ext.x` ו- `ext.y`. ככל שיש הנקיות `entry` או `extern`. בקובץ המקור.
נציג כתע את הפורמטים של קבצי הפלט. דוגמאות יובאו בהמשך.

פורמט קובץ ה- object

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

כזכור, האסמבלר מקודד את ההוראות כך שתמונה ההוראות תתאים לטיענה החל מכתובת 100
(עשורוני) בזיכרון. נשים לב שرك המעבר הראשון יודעים מהו הגול הכלול 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 המעבר
הראשון, את ערכי הסמלים המאופינים כ-`data` (כזכור, באлогריתם השLAG עלי, בצעד 19,
הוספנו לכל סמל כזה את הערך ICF). במעבר השני, בהשלמת הקיזוד של مليות-המידע,
משתמשים בערכים המעודכנים של הסמלים, המותאמים למבנה המלא והסופי של תמונה הזיכרון.

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

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

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

פורמט קובץ ה- entries

קובץ-`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 ארבע
ספרות, כולל אפסים מובילים). בין שני השדות בשורה יש רווח אחד. אינו חשיבות לסדר השורות,
כי כל שורה עומדת בפני עצמה.

פורמט קובץ ה- externals

קובץ-`externals` בניית אפ' הוא משורות טקסט, שורה לכל כתובות בקוד המcona בה יש مليות
מידע המתיחסת לסמל שמאופיין כ-`external`. כזכור, רשימה של مليות-מידע אלה נבנתה
במעבר השני (צעד 6 באлогריתם השLAG).

כל שורה בקובץ ה-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.

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

נדגים את הפלט שמייצר האסמבולר עבור קובץ מקור בשם **ps.as** הנתנו 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	001010101101	A
0101		Register r3	000000001000	A
0102		Address of label LIST	000010001000	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	111111111010	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 ps.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 לאסטמבלר אינו ידוע מראש, ולכן גם גודלו של קוד המוכנה אינו צפוי מראש. אולם בכך להקל בימוש האסטמבלר, מותר להניח גודל מקסימלי. לפיכך יש אפשרות להשתמש במערכות לאחסן תומנות קוד המוכנה בלבד. כל מבנה נתונים אחר (למשל טבלת הסמלים), יש למשה באופן ייעיל וחסכוני (למשל באמצעות רשיימה מקושרת והקצאת זיכרון דינמי).
- השמות של קבצי הפלט צריכים להיות תואמים לשם קובץ הקלט, למעט הסיומות. למשל, אם קובץ הקלט הוא as,prog אז קבצי הפלט שייצרו הם : prog.ob, prog.ext, prog.ent.
- מתכונת הפעלת האסטמבלר צריכה להיות כפי הנדרש בממ"ן, ללא שינוים כלשהם. ככלומר, ממש המשמש יהיה אך ורק באמצעות שורת הפוקודה. בפרט, שמורות קבצי המקור יועברו לתוכנית האסטמבלר כארגומנטים (אחד או יותר) בשורת הפוקודה. אין להוסיף תפריטי קלט אינטראקטיביים, חלונות גרפיים למיניהם, ועוד'.
- יש להקפיד לחלק את שימוש האסטמבלר למספר מודולים (קבצים בשפט C) לפי משימות. אין לרכז משימות מסוימים שונים במודול יחיד. מומלץ לחלק למודולים כגון : מעבר ראשון, מעבר שני, פונקציות עזר (למשל, תרגום לבסיס, ניתוח תחבירי של שורה), טבלת הסמלים, מפת הזיכרון, טבלאות קבועות (קוד הפעולה, שיטות המיעון החוקיקות לכל פעולה, ועוד').
- יש להקפיד ולתעד את השימוש באופן מלא וברור, באמצעות הערות מפורטות בקוד.
- יש לאפשר תווים לבנים.Undefים בקובץ הקלט בשפט אסטמבלרי. למשל, אם בשורת הוראה יש שני אופרנדים המופרדים בפסיק, אזי לפני ואחרי הפסיק מותר שייהיו רווחים וטאים בכל כמות. בדומה, גם לפני ואחרי שם הפעולה. מותר גם שורות ריקות. האסטמבלר יתעלם מהתווים לבנים מיותרים (כלומר ידלג עליהם).
- הקלט (קוד האסטמבלרי) עלול להכיל שגיאות תחביריות. על האסטמבלר לגלות ולדוח על כל השורות השגויות בקלט. אין לעצור את הטיפול בקובץ קלט לאחר גילוי השגיאה הראשונה. יש להדפיס לפחות הודעות מפורטות מכל 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).

גם ונשלם פרק ההסבירים והגדרת הפרויקט.

בשאלות ניתן לפנות ל专家组 הדין באתר הקורס, ועל כל אחד מהמנחים בשעות הקבלה שלהם.

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

لتשומתיכם : לא תיתנו דחיה בהגשת הממ"ן, פרט למקרים מיוחדים כגון מילאים או מחלת ממושכת. במקרים אלו יש לבקש ולקבל אישור מרחש מצוות הקורס.

בהצלחה !