

מטלת מנהה (ממ"ז) 14

הקורס : 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטרת הוראה : פרויקט גמר

משקל המטרת הוראה : 61 נקודות (חוובה)

מספר השאלות : 1

מועד אחרון להגשה : 11.08.2024

סמינר : 2024

קיימות שתי חלופות להגשת מטלות:

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס
 - שליחת מטלות באמצעות דואר אלקטרוני - **באישור המנהה בלבד**
- הסביר מפורט ב"נוהל הגשת מטלות מנהה"**

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

עליכם ל כתוב תוכנת אסמבלי, עבור שפת אסמבלי שתוגדר בהמשך. הפרויקט יכתב בשפת C.

עליכם להגיש את הפריטים הבאים:

1. קבצי המקור של התוכנית שתכתבם (קובציים בעלי סיומת .c או .h).
2. קובץ הרצה (מוקומפל ומקושר) עבור מערכת אוביונטו.
3. קובץ makefile. הקימפול חייב להיות עם הקומפיילר gcc והדגלים : -Wall -ansi -pedantic . יש לנפות את כל ההודעות שמצויה הקומפיילר, כך שהתוכנית תתקמפל ללא כל העוראות או זהירות.
4. דוגמאות הרצה (קלט ופלט):
 - א. קבצי קלט בשפת אסמבלי, ובקבצי הפלט שנוצרו מהפעלת האסמבולר על קבצי קלט אלה. יש להציג שימוש במגוון הפעולות וטיפוסי הנתונים של שפת האסמבלי.
 - ב. קבצי קלט בשפת אסמבלי המציגים מגוון רחב של סוגי שגיאות אסמבלי (ולכן לא נוצרים קבצי פלט), ותדריסי המשך המראים את ה Hodotutsgeshagia שמצויה האסמבולר.

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

זכיר מספר היבטים חשובים של כתיבת קוד טוב:

1. הפשטה של מבני הנתונים: רצוי (כל האפשר) להפריד בין הגישה למבנה הנתונים לבין המימוש 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 במסות משמעויות למשתנים ופונקציות. יש לעורך את הקוד באופן מסודר: הזחות עקביות, שורות ריקות להפרדה בין קטעי קוד, וכו'.
3. תיעוד: יש להכניס בקבצי המקור תיעוד תמציתי וברור, שיסביר את תפקידה של כל פונקציה (באמצעות העורות כותרת לכל פונקציה). כמו כן יש להסביר את תפקדים 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, על ציון גובה. כדי לקבל ציון גבוה, על התוכנית לעמוד בקריטריונים של כתיבה ותיעוד ברמה גבוהה, כמפורט לעיל, אשר משקל המשותף מגיע עד לכ- 40% משקל הפרויקט.

על המטלה להיות **מקורית לחולוטין**: אין להיעזר בספריות חיצונית מלבד הספריות הסטנדרטיות, וכמובן לא בקוד ולא בחלקיק קוד הנמצאים בראשת, במקור חיצוני וכו'.

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

מומלץ לקרוא את הגדרת הפרויקט פעם ראשונה ברצף, 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, עשוות לרוץ באותו 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 בוגש בזיכרונו, ונראה כמו רצף של ספרות ביןאריות. יחידת העיבוד המרכזית - היע"מ (CPU) - יודעת לפרק את הרץ הזה לקטעים קטנים בעלי משמעות: הוראות, מענים ונתונים.

למעשה, זיכרונו המחשב יכול הוא אוסף של סיביות, שנוהגים לראותן במקובצות ליחידות בעלות אורך קבוע (בתים, מילימ). לא ניתן להבחין, בעין שinaire מיוונת, בהבדל פיסי קלשו בין אותו חלק בזיכרון שבו נמצאת תוכנית לבין שאר הזיכרונו.

יחידת העיבוד המרכזית (היע"מ) יכולה לבצע מגוון פעולות פשוטות, הנקראות **הוראות מוכנה**, ולשם כך היא משתמשת ברגיסטרים (registers) הקיימים בתחום היע"מ, ובזיכרונו המחשב. **דוגמאות:** העברת מספר מתא בזיכרון לרגיסטר ביע"מ או בחרזה, הוספת 1 למספר הנמצא ברגיסטר, בדיקה האם מספר המאוחסן ברגיסטר שווה לאפס, חיבור וחיסור בין שני רגיסטרים, ועוד'.

הוראות המcona ושילובים שלן הן המרכיבות תוכנית כפי שהיא טעונה לזכרו בזמן ריצתה. כל תוכנית מקור (התוכנית כפי שנכתבה בידי המתכנן), תתרגם בסופו של דבר באמצעות תוכנה מיוחדת לצורה סופית זו.

היע"מ יודע לבצע קוד שנמצא בפורמט של **שפת מוכנה**. זהו רצף של ביטים, המהווים קידוד ביןארי של סדרת הוראות המכונה המרכיבות את התוכנית. קוד זהה אינו קרייא למשתמש, ולכן לא נוח לקובד (או לזרום) תוכניות ישירות בשפת מוכנה. **שפת אסמבלי (assembly language)** היא שפת תוכנות מאפשרת לייצג את הוראות המכונה בצורה סימבולית קלה ונוחה יותר לשימוש. כמובן שיש צורך לתרגם את הייצוג הסימבולי לקוד בשפת מוכנה, 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 מוכנה. האסambilר משמש בתפקיד דומה שפת אסambilי.

כל מודל של יע"מ (כולומר לכל אירוגן של מחשב) יש שפת מכונה ייודית משלו, בהתאם גם שפת אסambilי ייודית משלו. לפיכך, גם האסambilר (כלי התרגומם) הוא ייודוי 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 במאין זה.

המשימה בפרויקט זה היא לכתוב אסambilר (כולומר תוכנית המתרגם לשפת מוכנה), עבר שפת אסambilי שנגידר כאן במיוחד לצורך הפרויקט.

لتשומת לב: בהסבירים הכלליים על אופן עבודות תוכנת האסambilר, תהיה מדי פעם התייחסות גם לעבודת שלבי הקישור והטעינה. התייחסויות אלה נועדו על מנת לאפשר לכם להבין את המשך

תהליך העבודה של הפלט של תוכנת האסמבלר. אין לטעות: עליהם ל כתוב את תוכנית האסמבלר בלבד. אין ל כתוב את תוכניות הקישור והטעינה!!!

המחשב הדמיוני וסתת האסמבלי

נגיד לך את סתת האסמבלי ואת מודל המחשב הדמיוני, עבור פרויקט זה.
הערה: תיאור מודל המחשב להלן הוא חלקי בלבד, ככל שנחוץ לביצוע המשימות בפרויקט.

"חומרה":

המחשב בפרויקט מורכב ממעבד CPU (יע"מ - ייחידת עיבוד מרכזית), אוגרים (רגיסטרים) וזכרון RAM. חלק מהזיכרון משמש גם כמחסנית (stack).

למעבד 8 רגיסטרים כלליים, בשמות: r0, r1, r2, r3, r4, r5, r6, r7. גודלו של כל רגיסטר הוא 15 סיביות. הסיבית ה-1 הינה משמעותית תצון כסיבית מס' 0, והסיבית המשמעותית ביותר במס' 14. שמות הרגיסטרים נכתבים תמיד עם אות 'z' קטנה.

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

גודל הזיכרון הוא 4096 תאים, בכתביות 0-4095 (בסיס עשרוני), וכל תא הוא בגודל של 15 סיביות. לתא בזיכרון נקרא גם בשם "מילה". הסיביות בכל מילה ממושפרות כמו ברגיסטר.

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

מבנה הוראות מכונה:

כל הוראה מכונה מקודדת במספר מילوت זיכרון רצופות, החל מ Mills אחת ועד ל Mills שלוש Millions, בהתאם לשיטות המיעון בהן נעשה שימוש (ראו בהמשך).

בקובץ הפלט המכיל את קוד המכונה שבונה האסמבלר, כל מילה תקודד בסיס אוקטלי (ראו פרטים לגבי קבצי פלט בהמשך):

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

	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	שיטה מיעון				שיטה מיעון				שיטה מיעון				השדה		
	אופרנד מקור				אופרד יעד								A	R	E
opcode	ש	ש	ש	ש	ש	ש	ש	ש	ש	ש	ש	ש	ה	ט	ט
	י	י	י	י	י	י	י	י	י	י	י	י	ה	ט	ט
	ט	ט	ט	ט	ט	ט	ט	ט	ט	ט	ט	ט	ה	ט	ט
	ה	ה	ה	ה	ה	ה	ה	ה	ה	ה	ה	ה	ה	ט	ט
	3	2	1	0	3	2	1	0							

פירוט השיטות 3-0 מופיע בסעיף "שיטות מיעון" בהמשך.

בשפה שלנו יש 16 קודים פעולה והם :

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

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

סיביות 11-14: במילה הראשונה של הוראה סיביות אלה מהוות את **קוד-הפעלה** (opcode). כל opcode מיוצג בשפת אסמבלי באופן סימבולי על ידי **שם-פעולה**.

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

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

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

לשימוש לב : **השדה 'E,A,R'** מתווסף לכל אחת מהamilims בקידוד ההוראה (ראו פירוט של שיטות המיעון בהמשך).

שיטות מיעון :

בשפת האסמבלי שלנו קיימות ארבע שיטות מיעון, המסומנות במספרים 0,1,2,3. השימוש בשיטות מיעון מצריך קידוד של מילوت-מידע נוספת בקוד המכונה של כל הוראת מכונה. כל אופרנד של ההוראה דורש מילה אחת נוספת.

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

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

- סיבית 2 (הסיבית A) מצינית שקידוד המילה הוא מוחלט (Absolute), ומחייב שינוי בשלבי הקישור והطיענה.
- סיבית 1 (הסיבית R) מצינית שהקידוד הוא של כתובות פנימית הניתנת להזזה (Relocatable), ומחייב שינוי בשלבי הקישור והטיענה.
- סיבית 0 (הסיבית E) מצינית שהקידוד הוא של כתובות חיצונית (External) , ומחייב שינוי בשלבי הקישור והטיענה.

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

ערך השדה 'A,R,E' הנדרש בכל שיטת מיון מופיע בתיאור שיטות המיון להלן.

ערך	שיטת המיון	תוכן המילה הנוספת	אופן הכתיבה	דוגמה
0	מיון מיידי	밀ת-מידע נוספת של ההוראה מכילה את האופרנד עצמו, שהוא מספר שלם בסיסי עשרוני. ברוחב של 12 סיביות, השוכן בסיביות 3-14 של המילה. הסיביות 0-2 של מילת המידע הן השדה E. במיון מיידי, ערך הסיבית A הוא 1, ושתי הסיביות האחרות מאופסות.	האופרנד מתחליל בתו # ולאחריו ובצמוד אליו מופיע מספרשלם בסיסי עשרוני.	mov #1,r2 בדוגמה זו האופרנד הראשון של הפקודה הראשונית של המורה (אופרנד המקור) נתון בשיטת מיון מיידי. ההוראה כותבת את הערך 1- אל אוגר r2
1	מיון ישיר	밀ת-מידע נוספת של ההוראה מכילה כתובות בזיכרון. המילה כתובות זו בזיכרון היא האופרנד. הכתובת מיוצגת כמספר ללא סימן ברוחב של 12 סיביות, בסיביות 3-14 של מילת המידע. הסיביות 0-2 במלת המידע הן השדה E. במיון ישיר, ערך הסיביות אלה תלוי בסוג הכתובת הרשומה בסיביות 14-3. אם זהה כתובות שמייצגת שורה בקובץ המקור הנוכחי (כתובת פנימית), ערך הסיבית R הוא 1, ושתי הסיביות האחרות מאופסות. ואילו אם זהה כתובות שמייצגת שורה בקובץ מקור אחר של התוכנית (כתובת חיצונית), ערך הסיבית E הוא 1, ושתי הסיביות האחרות מאופסות.	האופרנד הוא <u>תוויות</u> שכבר הוגדרה, או שתוגדר בהמשך הקובץ. הוגדרה נעשית על ידי כתיבת תוויות בתחילת הנקיטת '.data' או '.string', או בתחילת הוראה של התוכנית, או באמצעות אופרנד של הנקיטת '.extern'.	השורה הבאה מגדרה את התווית x : x: .data 23 ההוראה : dec x מكتינה ב-1 את תוכן המילה שבכתבות x בזיכרון (ה"משתנה" x).

ערך	שיטת המיעון	תוכן המילה הנוספת	אופן הכתיבה	דוגמה
2	מיון אוגר עקיף	<p>שיטת מיון זו מושמת לגישה לזכורו באמצעות מצביע שנמצא באוגר. תוכן האוגר הוא כתובות בזיכרון, והמילה בכתובת זו בזיכרון היא האופrnd. הכתובות מיוצגת באוגר כמספר ללא סימן ברוחב של 15 סיביות.</p> <p>אם האופrnd הוא אופrnd יעד, מילת-מידע נוספת של הפוקודה תכיל בסיביות 5-3 את מספרו של האוגר שימוש במצביע. ואילו אם האוגר הוא אופrnd מקור, מספר מילת-הميدע הנוספת.</p> <p>הסיביות 0-2 של מילת המידע הן השדה A,R,E. במיון אוגר עקיף, ערך הסיבית A הוא 1, ושתי הסיביות האחרות מאפסות.</p> <p>אם בפוקודה יש שני אופrndים, וכל אחד מהם בשיטת מיון אוגר עקיף או בשיטת מיון אוגר ישיר (ראו בהמשך), שני האוגרים יחלקו מילת-מידע אחת משותפת, כאשר סיביות 3-5 יכולים את מספר אוגר היעד, וסיביות 6-8 את מספר אוגר המקור. השדה A,R,E יהיה כמפורט לעיל.</p> <p>סיביות במילת-הميدע שאינו בשימוש יכולים 0.</p>	<p>האופrnd מתחליל בתו * ולאחריו ובצמוד אליו מופיע שם של אוגר.</p>	<p>inc *r1 בדוגמה זו, ההוראה מגדילה ב-1 את תוכן המילה בזיכרון עליה מצביע האוגר 1, כלומר המילה שכתובתה נמצאת באוגר 1.</p> <p>דוגמה נוספת: mov *r1,*r2 בדוגמה זו, ההוראה מעתקה את תוכן המילה בזיכרון עליה מצביע האוגר 1 אל המילה בזיכרון עליה מצביע האוגר 2. שני האופrndים בדוגמה זו הם בשיטת מיון אוגר עקיף, ולכן שני האוגרים יקודדו במילת-מידע נוספת בפוקודה.</p>
3	מיון אוגר ישיר	<p>האופrnd הוא שם של אוגר. אם האוגר משמש כאופrnd יעד, מילת-מידע נוספת של הפוקודה תכיל בסיביות 5-3 את מספרו של האוגר. ואילו אם האוגר משמש כאופrnd מקור, מספר האוגר יקודד בסיביות 6-8 של מילת-הميدע.</p> <p>הסיביות 0-2 של מילת המידע הן השדה A,R,E. במיון אוגר ישיר, ערך הסיבית A הוא 1, ושתי הסיביות האחרות מאפסות.</p> <p>אם בפוקודה יש שני אופrndים, וכל אחד מהם בשיטת מיון אוגר ישיר או בשיטת מיון אוגר עקיף (ראו לעיל), שני האוגרים יחלקו מילת-מידע אחת משותפת, כאשר סיביות 3-5 יכולים את מספר אוגר היעד, וסיביות 6-8 את מספר אוגר המקור. השדה A,R,E יהיה כמפורט לעיל.</p> <p>סיביות במילת-הميدע שאינו בשימוש יכולים 0.</p>	<p>האופrnd הוא שם של אוגר.</p>	<p>clr r1 בדוגמה זו, ההוראה מאפסת את תוכן האוגר 1. דוגמה נוספת: mov r1,*r2 בדוגמה זו, ההוראה מעתקה את תוכן האוגר 1 אל המילה בזיכרון 2 אליה מצביע האוגר 2. אופrnd המקור 1 נตอน בשיטת מיון אוגר ישיר, ואופrnd היעד 2 בשיטת מיון אוגר עקיף, ולכן האוגרים יקודדו במילת-מידע נוספת אחת משותפת.</p>

הערה: מותר להתייחס לתווית עוד לפני שמחבירים עליה, בתנאי שהיא אכן מוחרת במקומות כלשהו בקובץ.

מפורט הוראות המכונה:

בטיור הוראות המכונה נשתמש במונח **PC** (קיצור של "Program Counter".) זהו רגיסטר פנימי של המעבד לא רגיסטר כללי, שמכיל בכל רגע נתון את כתובת הזיכרונו בה נמצאת ההוראה הנוכחית שמתבצעת (הכוונה תמיד לכתובת המילה הראשונה של ההוראה).

הוראות המכונה מתחולקות לשלווש קבוצות, לפי מספר האופרנדים הנדרשים לפעולה.

קבוצת ההוראות הראשונה:
אלו הן הוראות המקבלות שני אופרנדים.

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

הוראה	הסבר הפעולה	דוגמה	הסבר הדוגמה
mov	מבצעת העתקה של האופרנד הראשון, אופרנד המקור (source) אל האופרנד השני, אופרנד היעד (destination) בהתאם לשיטת המיעון.	mov A, r1	העתק את תוכן המשתנה A (המילה שבכתובת A בזיכרון) אל רגיסטר r1.
cmp	מבצעת "השוואה" בין שני האופרנדים שליה. אופן ההשוואה: תוכן אופרנד היעד (השני) מופחת מתוכן אופרנד המקור (הראשון), ללא שימרת תוצאה החישור. פעולה החישור מעדכנת דגל باسم Z ("דגל האפס") ברגיסטר הסטטוס (PSW).	cmp A, r1	אם תוכן המשתנה A זהה לתוכנו של רגיסטר r1 אז דגל האפס, Z, ברגיסטר הסטטוס (PSW) יודלק, אחרת הדגל יאפס.
add	אופרנד היעד (השני) מקבל את תוצאה החיבור של תוכן המשתנה A ותוכנו הנוכחי של r0.	add A, r0	רגיסטר r0 מקבל את תוכנת החיבור של תוכן המשתנה A ותוכנו הנוכחי של r0.
sub	אופרנד היעד (השני) מקבל את תוצאה החישור של אופרנד המקור (הראשון) מօפרנד היעד (השני).	sub #3, r1	רגיסטר r1 מקבל את תוכנת החישור של הערך 3 מתוכנו הנוכחי של הרגיסטר r1.
lea	lea הוא קיצור (ראשי תיבות) של load effective address מציבה את המعن זיכרונו המיצג על ידי התווית שבօפרנד הראשון (המקור), אל אופרנד היעד (הօפרנד השני).	lea HELLO, r1	הعن שמייצגת התווית HELLO מוצב לרגיסטר r1.

קבוצת ההוראות השנייה:

אלו הן ההוראות הדורשות אופרנד אחד בלבד. אופן הקידוד של האופרנד הוא כמו של אופרנד היעד בפקודה עם שני אופרנדים. במקרה זה, השדה של אופרנד המקור (סיביות 10-7) בambilת הראשונה בקידוד ההוראה הינו חסר משמעות, ולפיכך יכול אפסים.

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

הוראה	הסבר הפעולה	דוגמה	הסבר הדוגמה
clr	איפוס תוכן האופרנד	clr r2	r2 ← 0
not	היפוך ערכי הסיביות באופרנד (כל סיבית שערכה 0 תהפוך ל-1 ולהיפך : 1 ל-0).	not r2	r2 ← not r2
inc	הגדלת תוכן האופרנד באחד.	inc r2	r2 ← r2 + 1
dec	הקטנת תוכן האופרנד באחד.	dec C	C ← C - 1
jmp	קפיצה (הסתעפות) בלתי מותנית אל ההוראה שנמצאת במען המוצג על ידי האופרנד. כלומר, בתוצאה מביצוע ההוראה, מצביע התוכנית (PC) קיבל את ערך אופרנד היעד.	jmp LINE	PC ← LINE מצביע התוכנית מקבל את המופיע המיוצג על ידי התווית LINE, ולפיכך הפוקודת הבאה שתתבצע תיהה בمعן זה.
bne	אם ערך הדגל Z באוגר השטוטוס (PSW) הינו 0, אז : bne LINE זהי הוראת הסתעפות מותנית. מצביע התוכנית (PC) קיבל את ערך אופרנד היעד אם ערכו של הדגל Z באוגר השטוטוס (PSW) נקבע הינו 0. כזכור, הדגל Z בפקודת cmp.	bne LINE	asm عרך הדגל Z באוגר השטוטוס (PSW) הינו 0, או : PC ← LINE
red	קריאה שלתו מהקלט הסטנדרטי (stdin) אל האופרנד.	red r1	קוד האסציא של התו הנקרא מהקלט ייכנס לאוגר r1.
prn	הדפסת התו הנמצא באופרנד, אל הפלט הסטנדרטי (stdout).	prn r1	התו אשר קוד האסציא שלו נמצא באוגר r1 יודפס לפט הסטנדרטי.
jsr	קריאה לשגרה (סברותינה), מצביע התוכנית (PC) הנוכחית נדחף לתוך המחסנית שבזיכרונו המחשב, והאופרנד מוכנס ל-PC.	jsr FUNC	push(PC) PC ← FUNC

קבוצת ההוראות השלישייה:

אלו הן ההוראות ללא אופרנדים. קידוד ההוראה מורכב ממילה אחת בלבד. השדרות של אופרנד המקור ושל אופרנד היעד במילה הראשונה של ההוראה אינם בשימוש, ולפיכך יהיו מאופסים.

ההוראות השינויים לקבוצה זו הן : .rts, stop

הווראה	מצב הפעולה	דוגמה	הסבר הדוגמה
rts	בחזרה משיגרה. הערך שנמצא בראש המחסנית של המחשב מוצא מן המחסנית, ומוכנס אל מצביע התוכנית (PC).	rts	PC ← pop()
stop	עצירת ריצת התוכנית.		התוכנית עצרת

מבנה שפת האסמבלי :

תכנית בשפת אסמבלי בנויה **מماקרואים ומשפטים** (statements).

ماקרואים :

ماקרואים הם קטעי קוד הכלולים בתוכם משפטיים. בתוכנית ניתן להגדיר ماקרו ולהשתמש בו במקומות שונים בתוכנית. השימוש בmacro ממקום מסוים בתוכנית יגרום לפרישת המacro לאותו מקום.

הגדרת macro נעשית באמצעות המacro המצוין:

```
macr m_macr
  inc r2
  mov A,r1
endmacr
```

שימוש בmacro הוא פשוט אזכור שמו. למשל, אם בתוכנית במקום כלשהו כתוב:

```
m_macr
```

```
m_macr
```

בדוגמה זו, השתמשנו פעמיים בmacro m_macr, התוכנית לאחר פרישת macro תיראה כך:

```
inc r2
mov A,r1
```

```
inc r2
mov A,r1
```

התוכנית לאחר פרישת המאקרו היא התוכנית שהאסטמבלר אמור לתרגם.

הנחות והניחות לגבי מאקרו:

- אין במערכת הגדרות מאקרו מקוונות (אין צורך לבדוק זאת).
- שם של הוראה או הנחיה לא יכול להיות שם של מאקרו (יש לבדוק זאת).
- ניתן להניח שלכל שורת מאקרו בקוד המקור קיימת סגירה עם שורת endmacro (אין צורך לבדוק זאת).
- הגדרת מאקרו תהיה תמיד לפני הקרייה למאקרו (אין צורך לבדוק זאת).
- נדרש שהקדם-אסטמבלר ייצור קובץ עם הקוד המורחב הכלול פרישה של המאקרו (הרחבה של קובץ המקור המתואר בהמשך). "קובץ המקור המורחב" הוא "קובץ מקור" לאחר פרישת המאקרו, לעומת "קובץ מקור ראשוני" שהוא קובץ הקלט למערכת, כולל הגדרות המאקרים.

לסיכום, במאקרו יש לבדוק :

- (1) שם המאקרו תקין (אינו שם הוראה וכדומה)
 - (2) בשורת ההגדירה ובשורת הסיום אין תוים נוספים אם נמצאה שגיאה בשל פרישת המאקרו - אי אפשר לעבור לשלבים הבאים : יש לעזר להודיע על השגיאות ולועבור לקובץ המקור הבא (אם קיים).
- הערה : **שגיאות בגוף המאקרו** (אם יש) מוגלים בשלבים הבאים.

משפטים :

קובץ מקור בשפט אסטטלי מורכב משורות המכילות משפטיים של השפה, כאשר כל משפט מופיע בשורה **נפרדת**. כלומר, הפרדה בין משפט למשפט בקובץ המקור הינה באמצעות התווים 't' (רווחים וטאים). ניתן בשורה אין אף תוו (למעט התוו 't') (שורה חדשה).

אורכה של שורה בקובץ המקור הוא 80 תוים לכל היותר (לא כולל התוו 't').

יש ארבעה סוגי משפטיים (שורות בקובץ המקור) בשפט אסטטלי, והם :

סוג המשפט	הסביר כלל
משפט ריק	זהוי שורה המכילה אך ורק תוים לבנים (whitespace), כלומר רק את התווים 't' ו- ' ' (רווחים וטאים). ניתן בשורה אין אף תוו (למעט התוו 't'), כלומר השורה ריקה.
משפט הערת	זהוי שורה בה התו הראשון הינו ',' (נקודה פסיק). על האסטטל להעתלם לחנותין משורה זו.
משפט הנחיה	זהו משפט המנחה את האסטטל מה עליו לעשות כשהוא פועל על תוכנית המקור. יש מספר סוגים של משפטי הנחיה. משפט הנחיה עשוי לגרום להקצאת זיכרון ואותחול משתנים של התוכנית, אך הוא אינו מייצר קידוד של הוראות מכונה המיעדות לביצוע בעת ריצת התוכנית.
משפט הוראה	זהו משפט המייצר קידוד של הוראות מכונה לביצוע בעת ריצת התוכנית. המשפט מורכב ממש של הוראה שעל המעבד לבצע, ותיאור האופרנדים של הוראה.

cut נפרט יותר לגבי סוגי המשפטים השונים.

משפט הנחיה :

משפט הנחיה הוא בעל המבנה הבא :

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

לאחר מכן מופיע שם הנחיה. לאחר שם הנחיה יופיעו פרמטרים (מספר הפרמטרים בהתאם לנחיה). שם של הנחיה מתחילה בתו ',' (נקודה) ולאחריו תוים באותיות קטנות (lower case) בלבד.

יש לשים לב: למיללים בקוד המכונה הנוצרות ממפט הוכח לא מצורף השדה E,R,A, והקידוד מלא את כל הסיביות של המילה.

יש ארבעה סוגים של משפטי הוכח, והם :

1. הוכח' .data

הפרמטרים של הוכח' .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 וביין פסיק למספר יכולים להופיע רווחים וטאים בכל כמות (או בכלל לא), אולם הפסק חייב להופיע בין המספרים. כמו כן, אסור שיופיע יותר מפסק אחד בין שני מספרים, וגם לא פסיק אחריו המספר האחרון או לפני המספר הראשון.

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

כלומר אם נכתב :

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

אז יוקצו בתמונת הנתונים ארבע מיללים רצופות שיכילו את המספרים שמופיעים בהוכח' .data XYZ מזויה עם כתובות המילה הראשונה.

אם נכתב בתכנית את ההוראה :

mov XYZ, r1

אז בזמן ריצת התכנית יוכנס לרגיסטר r1 ערך 7.

ואילו ההוראה :

lea XYZ, r1

תכניס לרגיסטר r1 את ערך התווית XYZ (כלומר הכתובת בזיכרון בה מאוחסן הערך 7).

2. הוכח' .string

הוכח' .string פרמטר אחד, שהוא מחרוזת חוקית. תוכי המחרוזת מקודדים לפי ערכי ה-ascii המתאימים, ומוכנסים אל תומנת הנתונים לפי סדרם, כל תו ב밀יה נפרד. בסוף המחרוזת יתווסף התן '0' (הערך המספרי 0), המסמן את סוף המחרוזת. מונחה הנתונים של האSEMBLER יקודם בהתאם לאורך המחרוזת (בתוספת מקום אחד עבור התו המסיים). אם בשורת ההוכח' מוגדרת תווית, אז תווית זו מקבלת את ערך מונחה הנתונים (לפני הקידום) ומוכנסת אל טבלת הסמלים, בדומה למה שנעשה עבור 'data'. (כלומר ערך התווית יהיה הכתובת בזיכרון שבתת-התחלה המחרוזת).

לדוגמה, הוכח' :

STR: .string "abcdef"

מקרה בתמונה הנתונים רצף של 7 מילימ', ומתחילה את המילימ' לקוד ascii של התווים לפי הסדר במחירות, ולאחריהםערך 0 לסימון סוף מחירות. התווית STR מזוהה עם כתובת התחלה המחרוזת.

3. הנקיה 'entry'

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

לדוגמה, השורות:

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

מודיעות לאסמליל שאפשר להתייחס בקובץ אחר לתווית O HELLO המוגדרת בקובץ הנוכחי.

لتשומת לב: תווית המוגדרת בתחילת שורת entry. הינה חסרת משמעות והאסמליל מועלם מהתווית זו (אפשר שהאסמליל יוציא הודעה אחרת).

4. הנקיה 'extern'

לנקיה 'extern'. פרמטר והוא שם של תווית שאינה מוגדרת בקובץ המקור הנוכחי. מטרת ההוראה היא להודיע לאסמליל כי התווית מוגדרת בקובץ מקור אחר, וכי קוד האסמליל בקובץ הנוכחי עושה בתווית שימוש.

שים לב כי הנקיה זו תואמת להנקיה entry. המופיע 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 לממי'ן זה).

לדוגמה, משפט הנקיה 'extern'. התואם לשפט הנקיה 'entry' מהדוגמה הקודמת יהיה:

```
.extern    HELLO
```

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

لتשומת לב: תווית המוגדרת בתחילת שורת extern. הינה חסרת משמעות והאסמליל מועלם מהתווית זו (אפשר שהאסמליל 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 ההוראה, אז היא תוכנס אל טבלת הסמלים. ערך התווית יהיה מען המילה הראשונה של ההוראה בתוך תMOVNT הקוד שבונה האסמליל.

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 טאבים (אחד או יותר).

כאשר יש שני אופרנדים, האופרנדים מופרדים זה מזה בתו ',' (פסיק). בדוגמה להנחתה 'data,' לא **חייבת להיות הצמדה של האופרנדים לפסיק**. כל כמות של רווחים ו/או טאבים משני צידי הפסיק היא חוקית.

למשפט הוראה עם שני אופרנדים המבנה הבא :

label: opcode source-operand, target-operand

לדוגמה :

HELLO: add r7, B

למשפט הוראה עם אופרנד אחד המבנה הבא :

label: opcode target-operand

לדוגמה :

HELLO: bne XYZ

למשפט הוראה ללא אופרנדים המבנה הבא :

label: opcode

לדוגמה :

END: stop

אפיון השדות במשפטים של שפת האסמבלי

תווית:
בתיאור שיטות המיעון מעלה הסברנו כי תווית היא ייצוג סימבולי של כתובות בזיכרון. נרחיב כאן את ההסבר :
תווית היא למעשה סמל שמודרך בתחילת המשפט הוראה, או בתחילת הנקיטת.data.string. או תווית חוקית מתחילה באות אלפביתית (גדולה או קטנה), ולאחריה סדרה כלשהי של אותיות אלפביתיות (גדולות או קטנות) ו/או ספרות. האורך המקסימלי של תווית הוא 31 תווים.

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

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

לדוגמה, התוויות המוגדרות להלן הן תוויות חוקיות.

hEllo:

x:

He78902:

لتשומת לב : מילים שמורות של שפת האסמבלי (כלומר שם של פעולה או הנחיה, או שם של רגייסטר) אינן יכולות לשמש גם שם של תווית. כמו כן, אסור שאותו סמל ישמש הן כתווית והן שם של מקאו (יש לבדוק זאת).

התוויות מקבלת את ערכה בהתאם להקשר בו היא מוגדרת.תוויות המוגדרת בהנחיות `.data`,
תקבל את ערך מונה הנטונים (data counter) הנווכי, בעוד שתוויות המוגדרת בשורת
הוראה תקבל את ערך מונה ההוראות (instruction counter) הנווכי.

لتשומת לב: מותר במשפט הוראה להשתמש באופrnd שהוא סמל שאינו מוגדר כתוית בקובץ
הנווכי, כל עוד הסמל מופיע כחיצוני (באמצעות הנחיה מודול). כלשיי בקובץ הנווכי.

מספר:

מספר חוקי מתחילה בסימן אופציוני: ‘-’, או ‘+’, ולאחריו סדרה כלשהי של ספרות בסיס עשרוני.
לדוגמא: `-5`, `76`, `+123` הם מספרים חוקיים. אין תמייחת השפט האסמבלי שלנו ביצוג בסיס
אחר מאשר עשרוני, ואין תמייחת במספרים שאינם שלמים.

מחרוזות:

מחרוזות חוקית היא סדרת תווי ascii נראים (שניתנים להדפסה), המוקפים במרקאות כפולות
(המרקאות אינן נחשבות חלק מהמחרוזת). דוגמה למחרוזות חוקית: “hello world”.

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

בכל מילה בקוד המכונה של הוראה (לא של נתוני), האסמבller מכניס מידע עבור תהליך הקישור
והטעינה. זה השדה A,R,E. המידע ישמש לתיקונים בקוד בכל פעם שיתיען לזכור לצורך הרצה.
האסמבller בונה מלכתחילה קוד שמיועד לטעינה החל מכתובת ההתחלה. התיקונים יאפשרו
טוען את הקוד בכל פעם למקום אחר, בלי צורך לחזור על תהליך האסמבller.

שלוש הסיביות בשדה E,A,R יכilo ערכאים בינאריים כפי שהסביר בתיאור שיטות המיעון.
המשמעות של כל ערך מפורטת להלן.

האות ‘A’ (קייזר של absolute) בא להציג שתוכן המילה אינו תלוי במקומות בו זיכרנו בו יייען
בפועל קוד המכונה של התכנית בעת ביצועה (למשל מילה המכילה אופrnd מיידי).

האות ‘R’ (קייזר של relocatable) בא להציג שתוכן המילה תלוי במקומות בו זיכרנו בו יייען בפועל קוד
המכונה של התכנית בעת ביצועה (למשל מילה המכילה כתובת שלתוויות המוגדרת בקובץ המקור).

האות ‘E’ (קייזר של external) בא להציג שתוכן המילה תלוי בערכו של סמל חיצוני (external)
(למשל מילה המכילה כתובת שלתוויות חיצונית, כלומר כתובת שאינה מוגדרת בקובץ המקור).

כאשר האסמבller מקבל תוכנית בשפת אסמבלי, עליו לטפל תחילת בפרישת המאקרוואים,
ורק לאחר מכן לעבור על התוכנית אליה נפרשו המאקרוואים. כמובן, פרישת המאקרוואים תעשה
בשלב “קדם אסמבller”, לפני שלב האסמבller (המתוואר בהמשך).

אם התכנית אינה מכילה מאקרו, תוכנית הפרישה תהיה זהה לתכנית המקורי.

דוגמה לשלב קדם אסמבller. האסמבller מקבל את התוכנית הבאה בשפת אסמבלי :

```
MAIN:    add   r3, LIST
LOOP:    prn   #48
          macr m_macr
          cmp   r3, #-6
          bne   END
          endmacr
          lea   STR, r6
          inc   r6
          mov   *r6,K
          sub   r1, r4
          m_macr
```

```

        dec   K
        jmp   LOOP
END:    stop
STR:    .string "abcd"
LIST:   .data 6, -9
        .data -100
K:      .data 31

```

תחילה האסמבילר עובר על התוכנית ופורש את כל המאקרואים הקיימים בה. רק אם תהליך זה מסתיים בהצלחה, ניתן לעبور לשלב הבא. אחרת, יש להציג את השגיאות ולא לייצר קבצים. בדוגמה זו, התוכנית לאחר פרישת המאקרו תיראה כך:

```

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

```

קוד התוכנית, לאחר הפרישה, ישמր בקובץ חדש, כפי שIOSVER בהמשך.

אלגוריתם שלדי של קדם האסמבילר

נציג להלן אלגוריתם שלדי לתהليل קדם האסמבילר. لتשומת לב: אין חובה להשתמש דווקא באלגוריתם זה:

1. קרא את השורה הבאה מקובץ המקור. אם נגמר הקובץ, עבור ל- 9 (סיום).
2. האם השדה הראשון הוא שם מאקרו בטבלת המאקרו (כגון m_macr)? אם כן, החלף את שם המאקרו והעתק במקומו את כל השורות המתאימות מהטבלה לקובץ, חוזר ל- 1. אחרת, המשך.
3. האם השדה הראשון הוא "macr" (התחלת הגדרת מאקרו)? אם לא, עבור ל- 6.
4. הדלק דגל "יש macr".
5. (קיימת הגדרת מאקרו) הכנס לטבלת שורות מאקרו את שם המאקרו (לדוגמה m_macr).
6. קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עבור ל- 9 (סיום).
7. האם זוהתה תווית endmacr הכנס את השורה לטבלת המאקרו ומחק את השורה הניל' מהקובץ. אחרת (לא מאקרו) חוזר ל- 1.
8. כבה דגל "יש macr". חוזר ל- 1. (סיום שמירת הגדרת מאקרו).
9. סיום: שמירת קובץ המאקרו פרוש.

אסמבלי עם שני מעברים

במעבר הראשון של האסמבלי, יש לזהות את הסמלים (תוויות) המופיעים בתוכנית, ולתת לכל סמל ערך מסווני שהוא המعنן בזיכרונו שהSAMPLE מיצג. במעבר השני, באמצעות ערכי הסמלים, וכן קודי-הפעולה ומספריו הריסטוריים, בונים את קוד המcona.

עליו להחליף את שמות הפעולות `mov, jmp, prn, sub, cmp, inc, bne, stop` בקוד הבינארי השקול להם במודל המחשב שהגדנו.

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

נניח שקטע הקוד לעיל (הוראות ונתונים) ייטע בזיכרון החל ממ عن 100 (בסיס 10). במקרה זה קיבל את ה"תרגום" הבא:

Decimal Address	Source Code	Explanation	Binary Machine Code
0100	MAIN: add r3, LIST	First word of instruction Source register 3 Address of label LIST	001010000010100 000000011000100 000010000010010
0103	LOOP: prn #48	Immediate value 48	1100000000000100 000000110000100
0105	lea STR, r6	Address of label STR Target register 6	010000101000100 00000111101010 000000000110100
0108	inc r6	Target register 6	011100001000100 000000000110100
0110	mov *r6,K	Source register 6 Address of label K	000001000010100 000000110000100 000010000101010
0113	sub r1, r4	Source register 1 and target register 4	001110001000100 000000001100100
0115	cmp r3, #-6	Source register 3 Immediate value -6	000110000001100 000000011000100 11111111010100
0118	bne END	Address of label END	101000000010100 000001110010101
0120	dec K	Address of label K	011100000010100 000010000001010
0122	Address of label LOOP	100100000010100 000001100111010	
0124	END: stop		111100000000100
0125	STR: .string "abcd"	Ascii code 'a'	000000001100001
0126		Ascii code 'b'	000000001100010
0127		Ascii code 'c'	000000001100011
0128		Ascii code 'd'	000000001100100
0129		Ascii code '\0' (end of string)	0000000000000000
0130	LIST: .data 6, -9	Integer 6 Integer -9	0000000000000110 111111111110111
0132	.data -100	Integer -100	111111110011100
0133	K: .data 31	Integer 31	000000000011111

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

כדי לבצע הממרה לבינארי של אופרנדים שכחובים בשיטות מעיון המשתמשים בסמלים (תוויות), יש צורך לבנות טבלה המכילה את ערכיו כל הסמלים. אולם בהבדל מהקודים של הפעולות, הידיעות מראש, הרי המענינים בזיכרונו עבור הסמלים ששימוש התוכנית אינם ידועים, עד אשר תוכנית המקור נסקרה כולה ונתגלו כל הגדרות הסמלים.

למשל, בקוד לעיל, האסטמבלר אינו יכול לדעת שהסמל END משוויך למן 124 (עשרוני), וההסמל K משוויך למן 133, אלא רק לאחר שנקרוו כל שורות התוכנית.

לכן מפרידים את הטיפול של האסטמבלר בסמלים לשני שלבים. בשלב הראשון בונים טבלה של כל הסמלים, עם הערכים המספריים המשויכים 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 בזיכרונו, ובה לכל סמל שבתוכנית המקור משוויך ערך מספרי, שהוא מען בזיכרונו. בדוגמה לעיל, טבלת הסמלים לאחר מעבר ראשון היא:

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

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

لتשומת לב: תפקיד האסטמבלר, על שני המעברים שלו, לתרגם קובץ מקור לקוד בשפת מכונה. בגמר פעולה האסטמבלר, התוכנית טרם מוכנה לטיעינה לזכרונו לצורך ביצוע. קוד המכונה חייב לעبور לשלב הקישור/טיעינה, ורק לאחר מכן לשלב הביצוע (שלבים אלה אינם חלק מהממ"ז).

המעבר הראשון

במעבר הראשון נדרשים כללים כדי לקבוע איזה מען ישוויך לכל סמל. העיקרונות הבסיסי הוא לספור את המיקומות בזיכרונו, אותן תופסות ההוראות. אם כל הוראה תיתען בזיכרונו למקום העוקב להוראה הקודמת, תציין ספרה כזאת את מען ההוראה הבאה. הספרה נעשית על ידי האסטמבלר ומוחזקת במונח ההוראות (IC). ערכו ההתחלתי של IC הוא 100 (עשרוני), וכן קוד המכונה של ההוראה הראשונה נבנה כך שייטען לזכרונו החל ממן 100. ה-IC מתעדכן בכל שורת הוראה המקצת מקום בזיכרונו. לאחר שעשה אסטמבלר קובע מהו אורך ההוראה, ה-IC מוגדל במספר התאים (מיילים) הנתפסים על ידי ההוראה, וכך הוא מצביע על התא הפניו הבא.

כאמור, כדי לקודד את ההוראות בשפת מכונה, מחזיק האסטמבלר טבלה, שיש בה קוד מתאים לכל שם פעולה. בזמן התרגום מחליף האסטמבלר כל שם פעולה בקוד שלו, וכן כל אופרנד מוחלף בקידוד מתאים, אך פעולה ה恰恰פה זו אינה כה פשוטה. ההוראות משתמשות בשיטות מייען מגוונות לאופרנדים. אותה פעולה יכולה לקבל שימושויות שונות, בכל אחת משתנות המייען, וכן יכולה להשתנות בהתאם לתא זיכרונו לרגעיסטר, או להעתיקת תוכן רגיסטר לרגיסטר אחר, וכן הלאה. ככל אפשרות כזו שלptom עשוי להתאים קידוד שונה.

על האסטמבלר לסרוק את שורת ההוראה בשלמותה, ולהחליט לגבי הקידוד לפי האופרנדים. בדרך כלל מתחלק הקידוד לשדה של שם הפעולה, ושדות נוספים המכילים מידע לגבי שיטות המייען. כל השדות ביחסים מילה אחת או יותר בקוד המכונה.

כאשר נתקל האסטבר בთווית המופיעה בתחילת השורה, הוא יודע שלפנוי הגדרה של תווית, ואז הוא משייך לה מען – תוכנו הנוכחי של ה-IC. כך מקבלות כל התוויות את מענהן בעת ההגדירה. תוויות אלה מוכנסות לטבלת הסמלים, המכילהinus, לשם התווית גם את המען ומאפיינים נוספים. כאשר תהיה התייחסות לთווית אופרנד של הוראה כלשהי, יוכל האסטבר לשולף את המען המתאים מטבלת הסמלים.

הוראה יכולה להתיחס גם לסמל שטרם הוגדר עד כה בתכנית, אלא יוגדר רק בהמשך התכנית. להלן דוגמה, הוראת הסתעפות למען שמווגדר על ידי התווית A שמשמעותה רק בהמשך הקוד :

bne A
•
•
•
A:

כאשר מגיע האסטבר לשורת ההסתעפה (bne A), הוא טרם נתקל בהגדרת התווית A ומכובן לא יודע את המען המשויך לתווית. לכן האסטבר לא יוכל לבנות את הקידוד הבינארי של האופרנד A. נראה בהמשך כיצד נפתרת בעיה זו.

בכל הוראה, תמיד אפשר לבנות בעבר הראשון את הקידוד הבינארי המלא של המילה הראשונה של כל הוראה, את הקידוד הבינארי של מילת-המידע הנוספת של אופרנד מיידי, או רגיסטר, וכן את הקידוד הבינארי של כל הנתונים (המתקבלים מההנחיות .string, .data, .).

המעבר השני

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

שם כך מבצע האסטבר מעבר נוסף (מעבר שני) על כל קובץ המקור, וمعدכן את קוד המcona של האופרנדים המשמשים בסמלים, באמצעות ערכי הסמלים מטבלת הסמלים. בסוף המעבר השני, תהיה התוכנית מתורגמת בslashes לקוד מכונה.

הפרדת הוראות ונתוניים

בתכנית מבחינים בשני סוגי של תוכן: הוראות ונתוניים. יש לארגן את קוד המcona כך שתתיה הפרדה בין הנתונים וההוראות. הפרדת ההוראות והנתוניים לקטעים שונים בזיכרון היא שיטה עדיפה על פני הצמדה של הגדרות הנתוניים להוראות המשמשות בהן.

אחד הסכנות הטමונות באירוע ההפרדת ההוראות מהנתוניים היא, שלפעמים עלול המעבד, בעקבות שגיאה לוגית בתכנית, לנוטות "לבצע" את הנתוניים כאילו היו הוראות חוקיות. למשל, שגיאה שיכולה לגרום תופעה זו הסתעפות לא נכון. התכנית כמובן לא תעבור נכון, אך לרוב הנזק הוא יותר חמור, כי נוצרת חריגת חומרה ברגע שהמעבד מבצע פעולה שאינה חוקית.

האסטבר שלנו חייב להפריד, בקוד המcona שהוא מיצר, בין קטע הנתוניים לקטע ההוראות. כולם בקובץ הפלט (בקוד המcona) תהיה הפרדה של הוראות ונתוניים לשני קטעים נפרדים, ואילו בקובץ הקלט אין חובה שתהייה הפרדה זו. בהמשך מתואר אלגוריתם של האסטבר, ובו פרטים כיצד לבצע את ההפרדה.

גילוי שגיאות בתכנית המקור

הנחת המטרה היא שאין שגיאות בהגדרות המקור, ולכן קדם האסטבר אינו מכיל שלב גילוי שגיאות, לעומת זאת האסטבר אמר לגלות ולדוח על שגיאות בתחביר של תוכנית המקור,

כגון פעולה שאינה קיימת, מספר אופרנדים שני, סוג אופרנד שאינו מתאים לפעולה, שם וגייסטר לא קיימים, ועוד שגיאות אחרות. כמו כן מוגדר האסטブル בכל סמל מוגדר פעמי אחת בדיק.

מכאן, שכל שגיאה המתגלת על ידי האסטブル נגרמת (בדרכן כלל) על ידי שורת קלט מסויימת.

לדוגמא, אם מופיעים שני אופרנדים בהוראה שאמור להיות בה רק אופרנד אחד, האסטブル ייתן הודעת שגיאה בנוסח "יותר מדי אופרנדים".

הערה: אם יש שגיאה בקוד האסטブル בוגר מאקרו, הרי שגיאת זו יכולה להופיע ולהתגלותשוב ושוב, בכל מקום בו נפרש המאקרו. נשים לב שכאשר האסטブル בודק שגיאות, כבר לא ניתן לזהות שזה קוד שנפרש ממקארו, כך שלא ניתן לחסוך גילויי שגיאה כפולים.

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

לתשומת לב: האסטブル אין עצור את פעולתו אחרי שנמצאה השגיאה הראשונה, אלא ממשיך לעבור על הקלט כדי לגלוות שגיאות נוספות, ככל שישן. כמובן שאין כל טעם לייצר את קבצי הפלט אם נתגלו שגיאות (ממילא אי אפשר להשלים את קוד המכונה).

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

שם פעולה	שיטות מיעון חוקיות עבור אופרנד מקור	שיטות מיעון חוקיות עבור אופרנד יעד
<code>mov</code>	0,1,2,3	1,2,3
<code>cmp</code>	0,1,2,3	0,1,2,3
<code>add</code>	0,1,2,3	1,2,3
<code>sub</code>	0,1,2,3	1,2,3
<code>lea</code>	1	1,2,3
<code>clr</code>	אין אופרנד מקור	1,2,3
<code>not</code>	אין אופרנד מקור	1,2,3
<code>inc</code>	אין אופרנד מקור	1,2,3
<code>dec</code>	אין אופרנד מקור	1,2,3
<code>jmp</code>	אין אופרנד מקור	1,2
<code>bne</code>	אין אופרנד מקור	1,2
<code>red</code>	אין אופרנד מקור	1,2,3
<code>prn</code>	אין אופרנד מקור	0,1,2,3
<code>jsr</code>	אין אופרנד מקור	1,2
<code>rts</code>	אין אופרנד מקור	אין אופרנד יעד
<code>stop</code>	אין אופרנד מקור	אין אופרנד יעד

אלגוריתם שלדי של האסטブル

לחיזוד ההבנה של תהליכי העבודה של האסטブル, נציג להלן אלגוריתם שלדי למעבר הראשון ולמעבר השני. لتשומת לב: אין חובה להשתמש דווקא באלגוריתם זה.

אנו מחלקים את קוד המכונה לשני אזוריים, אזור ההוראות (code) ואזור הנתונים (data). לכל אזור יש מונה משלה, ונסמן IC (МОНА ההוראות - Instruction-Counter - DC-1) ו- DC (МОНА הנתונים - Data-Counter). בניית קוד המכונה כך שיתאים לטעינה ל זיכרון החל מכתובת 100.

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

בכל מעבר מתחילה לקרוא את קובץ המקור מהתחלה.

מעבר ראשוני

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

מעבר שני

1. אתחל $0, IC \leftarrow 0$.
2. קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עברו ל-9.
3. האם השדה הראשון הוא סמל, דילג עליו.
4. האם זהה הינה `.data`. או `? .string`? אם כן, חזרו ל-2.
5. האם זהה הינה `.entry`? אם לא, עברו ל-7.
6. סמן בטבלת הסמלים את הסמלים המתאימים במאפיין `.entry`. חזרו ל-2.
7. השלים את קידוד האופרנדים החל מהמילה השנייה בקוד הבינארי של ההוראה, בהתאם לשיטת המיעון. אם אופרנד מכיל סמל, מצא את הערך בטבלת הסמלים (אם הסמל לא נמצא בטבלה, יש להודיע על שגיאיה).
8. עדכן $IC \leftarrow IC + L$, וחזרו ל-2.
9. קובץ המקור נקרא בשלמותו. אם נמצא שגיאות במעבר השני, עצור כאן.
10. בנה CUT קבוע הפלט (פרטים נוספים בהמשך).

נפעיל אלגוריתם זה על תוכנית הדוגמה שראינו לעיל **(לאחר שלב פרישת המאקרוואיט)**, ונציג את הקוד הבינארי שמתתקבל במעבר ראשון ובמעבר שני. להלן שוב תוכנית הדוגמה.

```
.MAIN:      add   r3, LIST
LOOP:       prn   #48
            lea    STR, r6
            inc    r6
            mov    *r6,K
            sub    r1, r4
            cmp    r3, #-6
            bne   END
            dec    K
```

```

        jmp  LOOP
END:    stop
STR:    .string "abcd"
LIST:   .data 6, -9
        .data -100
K:      .data 31

```

בוצע עתה מעבר ראשון על הקוד הנוכחי. נבנה את טבלת הסמלים. כמו כן, נבצע במעבר זה גם את קידוד כל הנתונים, וקידוד המילה הראשונה של כל הוראה. את החלקים שעדיין לא מתרגמים במעבר זה, נשאיר כמוות שהם (מסומנים ב- ? בדוגמה להלן).

אנו מניחים שהקוד ייטע לזכרו החל מהמ عن 100 (בסיס דצימלי).

Decimal Address	Source Code	Explanation	Binary Machine Code
0100	MAIN: add r3, LIST	First word of instruction	00101000010100
0101		Source register 3	000000011000100
0102		Address of label LIST	?
0103	LOOP: prn #48	Immediate value 48	1100000000000100
0104			000000110000100
0105	lea STR, r6	Address of label STR	010000101000100
0106		Target register 6	?
0107			000000000110100
0108	inc r6	Target register 6	011100001000100
0109			000000000110100
0110	mov *r6,K	Source register 6	000001000010100
0111		Address of label K	000000110000100
0112			?
0113	sub r1, r4	Source register 1 and target register 4	001110001000100
0114			000000001100100
0115	cmp r3, #-6	Source register 3	000110000001100
0116		Immediate value -6	000000011000100
0117			111111111010100
0118	bne END	Address of label END	101000000010100
0119			?
0120	dec K	Address of label K	011100000010100
0121			?
0122	jmp LOOP	Address of label LOOP	100100000010100
0123			?
0124	END: stop		1111000000000100
0125	STR: .string "abcd"	Ascii code 'a'	000000001100001
0126		Ascii code 'b'	000000001100010
0127		Ascii code 'c'	000000001100011
0128		Ascii code 'd'	000000001100100
0129		Ascii code '\0' (end of string)	000000000000000
0130	LIST: .data 6, -9	Integer 6	0000000000000110
0131		Integer -9	111111111110111
0132	.data -100	Integer -100	111111110011100
0133	K: .data 31	Integer 31	0000000000011111

טבלת הסמלים :

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

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

Decimal Address	Source Code	Explanation	Binary Machine Code
0100	MAIN: add r3, LIST	First word of instruction Source register 3 Address of label LIST	001010000010100
0101			000000011000100
0102			000010000010010
0103	LOOP: prn #48		1100000000000100
0104		Immediate value 48	000000110000100
0105	lea STR, r6		010000101000100
0106		Address of label STR	00000111101010
0107		Target register 6	000000000110100
0108	inc r6		011100001000100
0109		Target register 6	000000000110100
0110	mov *r6,K		000001000010100
0111		Source register 6	000000110000100
0112		Address of label K	000010000101010
0113	sub r1, r4		001110001000100
0114		Source register 1 and target register 4	000000001100100
0115	cmp r3, #-6		000110000001100
0116		Source register 3	0000000011000100
0117		Immediate value -6	11111111010100
0118	bne END		101000000010100
0119		Address of label END	000001111001010
0120	dec K		011100000010100
0121		Address of label K	000010000001010
0122	jmp LOOP		100100000010100
0123		Address of label LOOP	000001100111010
0124	END: stop		111100000000100
0125	STR: .string "abcd"	Ascii code 'a'	000000001100001
0126		Ascii code 'b'	000000001100010
0127		Ascii code 'c'	000000001100011
0128		Ascii code 'd'	000000001100100
0129		Ascii code '\0' (end of string)	000000000000000
0130	LIST: .data 6, -9	Integer 6	0000000000000110
0131		Integer -9	111111111110111
0132	.data -100	Integer -100	111111110011100
0133	K: .data 31	Integer 31	000000000011111

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

קבצי קלט ופלט של האסמבילר

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

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

- קובץ .am, המכיל את קוד המקור לאחר שלב קדם האסמבילר (לאחר פרישת המאקרואים).
- קובץ .object, המכיל את קוד המוכונה.
- קובץ .externals, ובו פרטים על כל המקומות (הכתובות) בקוד המוכונה בהם יש מילת-מידע שמקודדת ערך של סמל שהוצהר בחיצוני (סמל שהופיע כאופרנד של הנחיתת .extern, ומופיע בביטול הסמלים כ-.external).
- קובץ .entries, ובו פרטים על כל סמל שימושה כ כניסה (סמל שהופיע כאופרנד של הנחיתת .entry, ומופיע בביטול הסמלים כ-.entry).

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

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

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

```
assembler x y hello
```

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

שמות קבצי הפלט מבוססים על שם קובץ הקלט, כפי שהופיע בשורת הפקודה, בתוספת סיומת מתאימה: הסיומת ".am". עברו קובץ לאחר פרישת מאקרו, הסיומת ".ob". עברו קובץ ה-object, הסיומת ".ent". עברו קובץ ה-entries, והסיומת ".ext". עברו קובץ ה-externals.

לדוגמה, בהפעלת האסמבילר באמצעות שורת הפקודה: assembler x יוצר קובץ פלט .ob.x, וכן קבצי פלט .ext.x ו-.entry.x. ככל שיש הנחיתות entry.או .extern. בקובץ המקור. אם אין מאקרו בקובץ המקור, אז קובץ ".am" יהיה זהה לקובץ ".as".

אופן פעולות האסמבילר

נרחיב כאן על אופן פעולות האסמבילר, בנוסף לאלגוריתם השימושי שניתן לעיל.

האסמבילר מחזיק שני מערכיים, שייקראו להלן מערך ההוראות ומערך הנתונים. מערכים אלו נתונים למשעה תמונה של זיכרון המוכנה (גודל כל כניסה בזיכרון זהה לגודלה של מילת מוכנה (בסיסיות). במערך ההוראות מכניס האסמבילר את הקידוד של הוראות המוכנה שנקרו בו מהלך המעבר על קובץ המקור. במערך הנתונים מכניס האסמבילר את קידוד הנתונים שנקרו מקובץ המקור (שורות מסווג .data.string).

לאסמבילר יש שני מונחים: מונה ההוראות (IC) ומונה הנתונים (DC). מונחים אלו מצביעים על המיקום הבא הפניו במערכות לעיל, בהתאם. שמתחל האסמבילר לעבר על קובץ מקור, שני מונחים אלו מואפסים.

בנוסף יש לאסמבילר טבלה, אשר בה נאספות כל התוויות בחן נתקל האסמבילר במהלך המעבר על הקובץ. לטבלה זו קוראים טבלת סמלים (symbol-table). לכל סמל (תווית) נשמרים שם, ערכו,

ומאפיינים שונים שצינו קודם, כגון המיקום (code או data) או אופן העדכון (external או relocatable).

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

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

2. שורת הוראה :

האסטבלר מוצא מהי הפעולה, ומהן שיטות המידע של האופרנדים. (מספר האופרנדים אותם הוא מחפש נקבע בהתאם להוראה אותה הוא מצא). האסטבלר קובע לכל אופרנד את ערכו באופן הבא :

- אם זה רגיסטר – האופרנד הוא מספר הרגייסטר.
- אם זו תווית (מידע ישיר) – האופרנד הוא ערך התווית כפי שמופיע בטבלת הסמלים (ייתכן והסמל טרם נמצא בטבלת הסמלים).
- אם זה המספר # ואחריו מספר – האופרנד הוא המספר עצמו.
- אם זו שיטת מידע אחרת – ערכו של האופרנד נקבע לפי המפרט של שיטת המידע (ראו תיאור שיטות המידע לעיל)

קביעת שיטת המידע נעשית בהתאם לתחביר של האופרנד, כפי שהסביר לעיל בהגדרת שיטות המידע. למשל, מספר מצין מידע, תווית מצינת מידע ישיר וכו'.

לאחר שהאסטבלר ניתח את השורה והחליט לגבי הפעולה, שיטת מידע אופרנד המקור (אם יש), ושיטת מידע אופרנד היעד (אם יש), הוא פועל בהתאם :

אם זה הפעולה בעלת שני אופרנדים, אז האסטבלר מכניס למערך ההוראות, במקומות עלייו מצביעו מונה ההוראות IC, את קוד המילה הראשונה של ההוראה (בשיטת הייצוג של ההוראות המכונה כפי שתואר קודם לכן). מילה זו מכילה את קוד הפעולה, ואת שיטת המידע. בנוסף "משרידי" האסטבלר מקום במערך עבור המילים הנוספות הנדרשות עבור הוראה זו, ומגדיל את מונה ההוראות בהתאם. אם אחד או שני אופרנדים הם בשיטת מידע רגייסטר או מיידי, האסטבלר מקובד כתת המילים הנוספות הרלוונטיות במערך ההוראות.

אם זה הפעולה בעלת אחד בלבד, כלומר אין אופרנד מקור, אז הקידוד הינו זהה לעיל, פרט לנסיבות של שיטת המידע של אופרנד המקור במילה הראשונה, אשר יכול תמיד 0, מכיוון שאין רלוונטיות לפעולה.

אם זה הפעולה ללא אופרנדים אז רק המילה הראשונה (והיחידה). הסיבות של שיטות המידע של שני אופרנדים יכולו 0.

אם בשורת ההוראה קיימת תווית, אז התווית מוכנסת אל טבלת הסמלים תחת השם המתאים, ערך התווית הוא ערך מונה ההוראות לפני קידוד ההוראה.

3. שורת הנחיה :

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

I. '.data' .
האסטבלר קורא את רשימת המספרים, המופיע לאחר '.data.', מכניס כל מספר אל מערך הנתונים, ומגדם את מצביע הנתונים DC באחד עבור כל מספר שהוכנס.

אם בשורה 'data.', יש תווית, אזי תווית זו מוכנסת לטבלת הסמלים. היא מקבלת את הערך של מונה הנתונים DC שלפני הכנסת המספרים למערך הנתונים. הטיפוס של התווית הוא relocatable, וכך גם מסומן שהגדירה ניתנה בחלק הנתונים.

בסוף המעבר הראשון,ערך התווית יעדכו בטבלת הסמלים על ידי הוספת ה-IC (כלומר הוספה הארוך הכלול של קידוד כל ההוראות). הסיבה לכך היא שבתמונה קוד המכונה, הנתונים מופרדים מההוראות, וכל הנתונים יופיעו אחרי כל ההוראות (ראו תיאור קבצי הפלט בהמשך).

II. 'string'

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

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

III. 'entry'

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

IV. 'extern'

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

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

בסוף המעבר הראשון, האסמבלר מעדכו בטבלת הסמלים כל סמל המופיע כ-'data', על ידי הוספת IC+100 (עשرون) לערכו של הסמל. הסיבה לכך היא שבתמונה הכלולה של קוד המכונה, הנתונים מופרדים מההוראות, וכל הנתונים נדרשים להופיע אחרי כל ההוראות. סמל מסווג data למקרה תווית באוצר הנתונים, והעדכו מוסף לערך הסמל (כלומר לכנותו בזיכרונו) את האורך הכלול של קידוד כל ההוראות, בתוספת כתובות התחלה הטעינה של הקוד, שהוא 100.

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

במעבר השני, האסמבלר מקודד באמצעות טבלת הסמלים את כל המילים במערך ההוראות שטרם קודדו במעבר הראשון. אלו הן מילים שצרכות להכיל כתובות של תוויות.

פורמט קובץ ה- object

האSEMBLER בונה את תומונת זיכרונו המcona כז שקידוד ההוראה הראשונה מקובץ האSEMBLER ייכנס למן 100 (בבסיס עשרוני) בזיכרונו, קידוד ההוראה השנייה יכנס למן העקב אחרי ההוראה הראשונה (תלו依 במספר המילים של ההוראה הראשונה), וכן הלאה עד להוראה الأخيرة.

מיד לאחר קידוד ההוראה الأخيرة, מכנים לתוכנת הזיכרונו את קידוד הנתונים שנבנו על ידי ההנחיות 'data.string'. הנתונים יוכנסו בסדר בו הם מופיעים בקובץ המקור. אופרנד של ההוראה שמתיחס לSAMPLE שהוגדר באותו קובץ, יקודד כך שיצביע על המקום המתאים בתומונת הזיכרונו שבונה האSEMBLER.

נשים לב שהמשתנים מופיעים בתומונת הזיכרונו אחרי ההוראות. זהה הסיבה בגללה יש לעדכן בטבלת הסמלים, בסוף המעבר הראשון, את ערכי הסמלים המגדירים נתונים (SAMPLE מסוג .data).

עקרונית, קובץ object מכיל את תומונת הזיכרונו שתוארכה כאן. קובץ object מורכב משורות של טקסט כדלקמן :

השורה הראשונה בקובץ ה- object היא "cotratt", המכילה שני מספרים (בבסיס עשרוני) : האורך הכולל של קטע ההוראות (במילוט זיכרונו) ואחריו האורך הכולל של קטע הנתונים (במילוט זיכרונו). בין שני המספרים יש רווח אחד.

השורות הבאות בקובץ מכילות את תומונת הזיכרונו. בכל שורה שני ערכים : כתובות של מילה בזיכרונו, ומfcn המילה. הכתובות תירשם בסיס עשרוני בארכיו ספרות (כוללAPSIM מוביילים). תוכן המילה יירשם בסיס אוקטלי ב-5 ספרות (כוללAPSIM מוביילים). בין שני הערכים בכל שורה יפריד רווח אחד.

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

פורמט קובץ ה- entries

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

פורמט קובץ ה- externals

קובץ ה-externals בניוי אף הוא משורות טקסט. כל שורה מכילה שם של SAMPLE שהוגדר `external`, וכתובת בקוד המcona בה יש קידוד המתייחס לSAMPLE זה. כМОון שיתכן ויש מספר כתובות בקוד המcona בהם מתיחסים לאותו SAMPLE חיצוני. לכל התייחסות כזו תהיה שורה נפרצת בקובץ ה-externals. הכתובות מיוצגות בסיס עשרוני.

נדגים את קבצי הפלט שמייצר האSEMBLER עבור קובץ מקור בשם ps.as שהודגם קודם לכן. התוכנית לאחר שלב פרישת המאקרו תיראה כך :

```
; file ps.as

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

להלן טבלת הקידוד המלא הבינארי שמתתקבל מקובץ המקור, ולאחריה הפורמטים קבצי הפלט השונים.

Decimal Address	Source Code	Explanation	Binary Machine Code
0100	MAIN: add r3, LIST	First word of instruction Source register 3 Address of label LIST	001010000010100 000000011000100 000010001001010
0101			
0102			
0103	jsr fn1		110100000010100
0104		Address of label fn1 (external)	000000000000001
0105	LOOP: prn #48		1100000000001100
0106		Immediate value 48	000000110000100
0107	lea STR, r6		010000101000100
0108		Address of label STR	000010000100010
0109		Target register 6	000000000110100
0110	inc r6		011100001000100
0111		Target register 6	000000000110100
0112	mov *r6, L3		000001000010100
0113		Source register 6	000000110000100
0114		Address of label L3 (external)	000000000000001
0115	sub r1, r4		001110001000100
0116		Source register 1 and target register 4	000000001100100
0117	cmp r3, #-6		000110000001100
0118		Source register 3	000000011000100
0119		Immediate value -6	11111111010100
0120	bne END		101000000010100
0121		Address of label END	000010000011010
0122	add r7, *r6		001010000100100
0123		Source register r0 and target register 6	00000011110100
0124	clr K		010100000010100
0125		Address of label K	000010001100010
0126	sub L3, L3		001100100010100
0127		Address of label L3 (external)	000000000000001
0128		Address of label L3 (external)	000000000000001
0129	jmp LOOP		100100000010100
0130		Address of label LOOP	000001101001010
0131	END: stop		111100000000100
0132	STR: .string "abcd"	Ascii code 'a'	000000001100001
0133		Ascii code 'b'	000000001100010
0134		Ascii code 'c'	000000001100011
0135		Ascii code 'd'	000000001100100
0136		Ascii code '\0' (end of string)	000000000000000
0137	LIST: .data 6, -9	Integer 6	000000000000110
0138		Integer -9	11111111110111
0139	.data -100	Integer -100	111111110011100
0140	K: .data 31	Integer 31	000000000011111

להלן תוכן קבצי הפלט של הדוגמה. הערכה : יש חשיבות לעימוד הנתונים בקבצים הקובץ ob :ps.

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

הקובץ ps.ent :
כל המספרים בבסיס עשרוני

הקובץ ps.ext :
כל המספרים בבסיס עשרוני

MAIN 100
LIST 137

fn1 0104
L3 0114
L3 0127
L3 0128

لتשומת לב : אם בקובץ המקור אין הנחיות extern. אז לא ייווצר קובץ ext. בדומה, אם אין בקובץ המקור הנחיות ..entry, לא ייווצר קובץ ent. אין ליזור קובץ ext או ent שנשאר ריק.
הערכה : אין חשיבות לסדר השורות בקבצים מסוג ent. או ext. כל שורה עומדת בפני עצמה.

סיכום והנחיות כלליות

- אורך התוכנית, הניתנת כקלט לאסטブル אין ידוע מראש, ולכן אורך התוכנית המתורגמת אין-Amor להיות צפוי מראש. אולם כדי להקל בימוש האסטブル, ניתן להניח גודל מקסימלי. לפיכך יש אפשרות להשתמש במערכות לאחסן תMOVNT קוד המכונה בלבד. כל מבנה נתונים אחר (למשל טבלת הסמלים וטבלת המאקרו), יש למשם באופן ייעיל וחסכוני (למשל באמצעות רשיימה מקושרת והקצתה זיכרון דינמי).
- השמות של קבצי הפלט צריכים להיות תואמים לשם קובץ הקלט, למעט הסיומות. למשל, אם קובץ הקלט הוא as, אז קבצי הפלט שיוצאו הם : prog.ob, prog.ext, prog.ent.
- מתכונת הפעלת האסטブル צריכה להיות כפי הנדרש בממ"ן, ללא שינוי קליהם. כלומר, ממשך המשמש יהיה אך ורק באמצעות שורת הפקודת. בפרט, שמות קבצי המקור יעברו לתכנית האסטブル כארגומנטים בשורת הפקודת. אין להוסיף תפריטי קלט אינטראקטיביים, חלונות גרפיים למיניהם, וכו'').
- יש להקפיד לחלק את מימוש האסטブル למספר מודולים (קבצים בשפת C) לפי משימות. אין לרכז משימות מסוימות 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, וכו'').
- יש להקפיד ולתעד את המימוש באופן מלא וברור, באמצעות העורות מפורטות בקוד.
- יש לאפשר תווים לבנים עודפים בקובץ הקלט בשפת אסטブル. 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. מותירות גם שורות ריקות. האסטブル יתעלם מתחומים לבנים מיותרים (כלומר ידלג עליהם).
- הקלט (קוד האסטブル) עלול להכיל שגיאות תחביריות. על האסטブル לגלות ולדוח על כל השורות השגויות בקלט. אין לעוצר את הטיפול בקובץ קלט לאחר גילוי השגיאה הראשונה. יש להדפיס למסך הודעות מפורטות ככל הצורך, כדי שאפשר יהיה להבין מה והיכן כל שגיאה. כמובן שם קובץ קלט מכיל שגיאות, אין טעם להפסיק עבورو את קבצי הפלט (ob, ext, ent).

טם ונשלם פרק ההסבירים והגדרות הפרויקט.

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

לחזיכיכם, 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.

لتשומתיכם : לא תינטע דחיה בהגשת הממ"ן, פרט לMKRIM חריגים במיוחד. במקרים אלו יש לבקש ולקלбел אישור מראש מנהלה הקבוצה.

בזה צלח !