

מטלת מנהה (ממ"י) 14

הקורס: 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטרת פרויקט גמר

מספר השאלות: 1

סמסטר: 2020א'

משקל המטרת: 31 נקודות (חוובה)

מועד آخرון להגשה: 29.3.2020

קיימות שתי חלופות להגשת מטלות:

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

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

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

עליכם להגיד :

1. קבצי המקור של התוכנית שתכתבם (קובציים בעלי סיווג C או h).
2. קובץ הרצה (מקומפל ומקושר) עבור מערכת אוביונטו.
3. קובץ makefile בкомפיילר gcc עם הדגמים : c-pedantic-Wall-ansi . יש לנפות את כל ההודעות שМОציא הקומפיילר, כך שהתוכנית תתקמפל ללא כל העורות או אזהרות.
4. דוגמאות הרצה (קלט ופלט) :

א. קובצי קלט בשפת אסמבלי, קובצי הפלט שנוצרו מהפעלת האסמבלי על קבצי קלט אלה. יש להציגים שימוש במגוון הפעולות וטיפוסי הנתונים של שפת האסמבלי.

ב. קובצי קלט בשפת אסמבלי המודגימים מגוון רחב של סוגי שגיאות אסמבלי (ולכן לא נוצרים קבצי פלט), ותדפסי המסן המראים את ה הודעות השגיאה שМОציא האסמבלי.

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

נזכיר מספר היבטים חשובים של כתיבת קוד טוב :

1. הפיטה של מבני הנתונים : רצוי (כל האפשר) להפריד בין הגישה למבנה הנתונים לבין הימוש של מבני הנתונים. כך, למשל, בעת כתיבת פונקציות לטיפול בטבלה, אין זה מעניינים של משתמשים בפונקציות אלה, האם הטבלה ממומשת באמצעות מערכ או באמצעות רישימה מקושרת.

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

3. תיעוד : יש להכניס קבצי המקור תיעוד תמציתי וברור, שיסביר את תפיקתה 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 שביצעת את הדריש ממנה, אינה לכשעצמה עรองה לעין גביה. כדי לקבל ציון גבוה, על התוכנית לעמוד בקריטריונים של כתיבה ותיעוד ברמה טובה, כמפורט לעיל, אשר משקל המשותף מגיע עד לכ- 40% משקל הפרויקט.

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

מומלץ לעבוד בזוגות. אין לעבוד בזוגותים 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 פעם ראשונה ברכז, לקבלת תמונה כללית לגבי הנדרש, ורק לאחר מכן לקרוא שוב בצורה מעמיקה יותר.

רקע כללי ומטרת הפרויקט

בידוע, קיימות שפות תוכנות רבות, ומספר גדול של תוכניות, הכתובות בשפות שונות, עשויות לרוץ באותו מחשב עצמו. כיצד "מכיר" המחשב כל כך הרבה השפות? התשובה פשוטה: המחשב מכיר, למעשה שפה אחת בלבד: הוראות ונתונים הכתובים בקודBINARI. קוד זה מהווים בשיכון, ונראה כמו רצף של ספרות בינאריות. יחידת העיבוד המרכזית - היעם (CPU) - יודעת לפרק את הרצף הזה לקטעים קטנים בעלי משמעות: הוראות, מענים ונתונים.

למעשה, זיכרונו המחשב כולל הוא אוסף של סיביות, שנוהגים לראותן כמקובצות ליחידות בעלות אורך קבוע (בתים, מילימטים). לא ניתן להבחין, בין שאיתו מיום נתן, בהבדל פיסי כלשהו בין אותן חלק בזיכרון שבו נמצאת תוכנית לבין שאר הזיכרון.

יחידת העיבוד המרכזית (היעם) יכולה לבצע מגוון פעולות פשוטות, הנקראות הוראות מוכנה, ולשם כך היא משתמשת בארגרים (registers) הקיימים בתחום היעם, ובזיכרון המחשב.
דוגמאות: העברת מספר מתא בזיכרון לאוגר ביעם או בזרה, הוספת 1 למספר הנמצא באוגר, בדיקה האם מספר המאוחסן באוגר שווה לאפס, חיבור וחיסור בין שני אוגרים, וכו'. הוראות המכונה ושילובים שליהן הן המרכיבות תוכנית כפיה טעונה לזכורו בזמן ריצתה. כל תוכנית מקור (התוכנית כפי שנכתבה בידי המתכנן), תתרגם בסופו של דבר באמצעות תוכנה מיוחדת לזרה טפית זו.

היעם יודע לבצע קוד שנמצא בפורמט של שפת מוכנה. זה רצף של ביטים, המהווים קידוד בינהריא של סדרת הוראות המכונה המרכיבות את התוכנית. קוד זה אינו קרייא למשתמש, ולא נוח לקוד (או לזרא) תכניות ישירות בשפת מוכנה. **שפת אסמבלי (assembly language)** היא שפת תוכנות מאפשרת לייצג את הוראות המכונה בצורה סימבולית קלה ונוחה יותר לשימוש. כМОבן שיש צורך לתרגם את הייצוג הסימבולי לקוד בשפת מוכנה, כדי שהתוכנית תוכל לרוץ במחשב. תרגום זה נעשה באמצעות כלי שנקרא אסמבלי (assembler).

בידוע, לכל שפת תוכנות עליית יש מהדר (compiler), או מפרש (interpreter), המתרגמים תוכניות מקור לשפת מוכנה. האסמבלי משמש בתפקיד דומה עבור שפת אסמבלי.

לכל מודל של יעם (כולומר לכל אירוגון של מחשב) יש שפת מוכנה יעודית משלו, ובהתאם גם שפת אסמבלי יעודית משלו. לפיכך, גם האסמבלי (כלי התרגום) הוא יעודית ושונה לכל יעם.

תפקידו של האסמבלי הוא לבנות קוד מכיל קוד מוכנה, מקובץ נתון של תוכנית הכתובת בשפת אסמבלי. זהו השלב הראשון במסלול אותו עברת התוכנית, עד לקבלת קוד המוכן לריצה על חומרה. השלבים הבאים הם קישור (linkage) וטעינה (loading), אך בהם לא נוטק במילוי זה. המשימה בפרויקט זה היא כתוב אסמבלי (כולומר תוכנית המתרגמת לשפת מוכנה), עבר שפת אסמבלי שנגדיר כאן במיוחד לצורך הפרויקט.

لتשומת לב: בהסברים הכלליים על אופן העבודה תוכנת האסמבלי, תהיה מדי פעם התייחסות גם לעובדות שלבי הקישור והטעינה. התייחסויות אלה נעדו על מנת לאפשר לכם להבין את המשך תהליך העיבוד של הפלט של תוכנת האסמבלי. אין לטעות: עליכם כתוב את תוכנית האסמבלי בלבד. אין כתוב את תוכניות הקישור והטעינה!!!

המחשב הדמיוני ושפת האסמבלי

נדיר עתה את שפת האסמבלי ואת מודל המחשב הדמיוני, עברו פרויקט זה.
הערה: תואר 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 כליליים,uşnames: r7, r6, r5, r4, r3, r2, r1, r0. גודלו של כל אוגר הוא 15 סיביות. הסיבית הכיפתית מכילה מספר דגלים המאפיינים את המשמעותית ביותר ביותם מס' 14. שמות האוגרים כתובים תמיד כסיבית מס' 0, והסיבית המשמעותית ביותר ביותם מס' 1. אוגר אחד מילא מושפרות כמו זאת 'z', קטנה.

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

וגול הזיכרון הוא 4096 תאים, בכתובות 4095-0 (בסיס עשרוני), וכל תא הוא בגודל של 15 סיביות. לא בזיכרון נקרא גם בשם "מיליה". הסיביות בכל מילה מושפרות כמו באוגר.

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

מבנה הוראות מכונה:

כל הוראה מכונה מקודדת למספר מיליות זיכרון רצופות, החל ממילה אחת ועד למקסימום שלוש מילים, הכל בהתאם לשיטות המיעון בהן נעשה שימוש (ראו בהמשך).

בקוצ הפלט המכיל את קוד המכונה שבונה האסמבלי, 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	
opcode	ש	ש	ש	ש	ש	ש	ש	ש	ש	ש	י	ט	ט	A	R	E
	,	,	,	,	,	,	,	,	,	,	,	,	,			
	ט	ט	ט	ט	ט	ט	ט	ט	ט	ט	ט	ה	ה			
	ה	ה	ה	ה	ה	ה	ה	ה	ה	ה	ה	ה	ה			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0

סיביות 11-14: במילה הראשונה של ההוראה סיביות אלה מהוות את **קוד-הפעולה** (opcode).

במודל המכונה שלנו יש 16 קודי פעולה. כל קוד-פעולה (opcode) מיוצג בשפת אסמבלי באופן סימבולי על ידי **שם-פעולה**.

שם הפעולה (בבסיס 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

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

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

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

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

לתשומת לב: השדה 'A' מתווסף לכל אחת מהamilims בקידוז ההוראה (ראו פירוט של שיטות המיעון בהמשך).

שיטות מיעון:

בשפה האסטטוצייל שילנו קיימות ארבע שיטות מיעון, המסווגות במספרים 0,1,2,3. השימוש בשיטות מיעון מצרך קידוד של מילט-מידע נוספת נספotta בקוד המכונה של כל הוראות מכונה. כל אופרנד של ההוראה דרוש מילה אחת נוספת.

כאשר בהוראה יש שני אופרנדים, קודם תופיע מילט-המידע הנוסף של האופרנד הראשון (אופרנד הממקור), ולאחריה מילט-המידע הנוסף של האופרנד השני (אופרנד היעד). קיימים גם מקרים מיוחדים בו קידוד שני האופרנדים נעשה באמצעות מילט-מידע אחד משותף לשני האופרנדים.

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

• סיבית 2 (הסיבית A) מצינית שקיידוד המילה הוא מוחלט (Absolute), ואינו נדרש שינוי.
בשלבי הקישור והטיענה.

• סיבית 1 (הסיבית R) מצינית שהקיידוד הוא של כתובת פנימית הניתנת להזיה (Relocatable), ומצריך שינוי בשלבי הקישור והטיענה.

• סיבית 0 (הסיבית E) מצינית שהקיידוד הוא של כתובת חיצונית (External), ומצריך שינוי.
בשלבי הקישור והטיענה.

הסבר על התפקיד של השדה 'A,R,E' בקוד המכונה יבוא בהמשך.

ערך השדה 'E,A,R' הנדרש בכל שיטת מיעון מופיע בתיאור שיטות המיעון להלן.

מספר	שיטת המיעון	תוכן מילת-המידע הנוספת	אופן כתיבת האופרנד	דוגמה
0	מיעון מיידי	밀ת-מידע נוספת של ההוראה מכילה את האופרנד עצמו, שהוא מספר שלם בשיטת המשלים ל-2, ברוחב של 12 סיביות, השוכן בסיביות 3-14 של המילה.	האופרנד מתחילה בתו # ולאחריו ובצמוד אליו מופיע מספר שלם בסיס עשרוני.	mov #-1,r2 בדוגמה זו האופרנד הראשון של הפקודה (אופרנד המקור) נתון בשיטת מיעון מיידי. ההוראה כתובת את הערך 1- אל אוגר r2
1	מיעון ישיר	밀ת-מידע נוספת של ההוראה מכילה כתובת בזיכרון. המילה כתובת זו בזיכרון היא האופרנד. הכתובת מוצגת כמספר לא סימני ברוחב של 12 סיביות, בסיביות 3-14 של מילת המידע.	האופרנד הוא <u>תוויות</u> שכבר הוגדרה, או שתוגדר בהמשך הקובץ. הכתובת נעשית על ידי כתיבת תווית בתחילת הנקית '.data', או '.', או בתחלת הוראה של התוכנית, או באמצעות אופרנד של הנקית 'extern'.	השורה הבאה מגדרה את התווית x: x: .data 23 ההוראה: dec x מקטינה ב-1 את תוכן המילה שבכתבות x בזיכרון ("משתנה" x).

מספר	שיטת המיעון	תוכנית מילת-המידע הנוספת	אופן כתיבת האופrnd	דוגמה
2	מייעון אוגר עקיף	<p>שיטות מייעון זו משמשת לגישה לזכרון באמצעות מצביע שמנצ'א באוגר. תוכן האוגר הוא כתובות בזיכרון, והמילה בכתבoted זו בזיכרון היא האופrnd. הכתובות מוצגת באוגר כמספר ללא סימן ברוחב של 15 סיביות.</p> <p>אם האופrnd הוא אופrnd יעד, מילת-מידע נוספת נסflat של הפוקודהiscal בסיביות 3-5 את מספרו של האוגר שימוש במצביע. ואילו אם האוגר הוא אופrnd מקור, מספר האוגר יקודד בסיביות 6-8 של מילת-המידע הנוספת.</p> <p>הסיביות 0-2 של מילת המידע הן השדה E,A,R,E. במייעון אוגר עקיף, ערך הסיבית A הוא 1, ושתי הסיביות האחרות מאופסות.</p> <p>אם בפוקודה יש שני אופrndים, וכל אחד מהם בשיטת מייעון אוגר עקיף או בשיטת מייעון אוגר ישיר (ראו בהמשך), שני האוגרים יחלקו מילת-מידע אחת משותפת, כאשר סיביות 3-5 יכילה את מספר אוגר העיד, וסיביות 6-8 את מספר אוגר המקור. השדה A,R,E יהיה כמפורט לעיל.</p> <p>סיביות במילת-המידע שאין בשימוש יכילה 0.</p>	<p>האופrnd מתחליל בתו * ולאחריו ובצמוד אליו מופיע שם של אוגר.</p>	<p>בדוגמה זו, ההוראה inc מגדילה ב-1 את תוכן המילה בזיכרון עליה מצביע האוגר 1, ככלומר המילה שתכתבה נמצאת באוגר 1.</p> <p>דוגמה נוספת: mov *r1,*r2</p> <p>בדוגמה זו, ההוראה mov מעתיקה את תוכן המילה בזיכרון עליה מצביע האוגר 1 אל המילה בזיכרון עליה מצביע האוגר 2.</p> <p>שני האופrndים בדוגמה זו הם בשיטת מייעון אוגר עקיף, ולכן יקודדו במילת-מידע נוספת אחת משותפת.</p>
3	מייעון אוגר ישיר	<p>האופrnd הוא שם של אוגר. אם האוגר משמש כאופrnd יעד, מילת-מידע נוספת של הפוקודהiscal בסיביות 3-5 את מספרו של האוגר. ואילו אם האוגר משמש כאופrnd מקור, מספר האוגר יקודד בסיביות 6-8 של מילת-המידע.</p> <p>הסיביות 0-2 של מילת המידע הן השדה E,A,R,E. במייעון אוגר ישיר, ערך הסיבית A הוא 1, ושתי הסיביות האחרות מאופסות.</p> <p>אם בפוקודה יש שני אופrndים, וכל אחד מהם בשיטת מייעון אוגר ישיר או בשיטת מייעון אוגר עקיף (ראו לעיל), שני האוגרים יחלקו מילת-מידע אחת משותפת, כאשר סיביות 3-5 יכילה את מספר אוגר העיד, וסיביות 6-8 את מספר אוגר המקור. השדה A,R,E יהיה כמפורט לעיל.</p> <p>סיביות במילת-המידע שאין בשימוש יכילה 0.</p>	<p>האופrnd הוא שם של אוגר.</p>	<p>בדוגמה זו, ההוראה clr1 מאפסת את תוכן האוגר 1.</p> <p>דוגמה נוספת: mov r1,*r2</p> <p>בדוגמה זו, ההוראה mov מעתיקה את תוכן האוגר 1 אל המילה בזיכרון 2 אליה מצביע האוגר 2.</p> <p>אופrnd המקור 1 נקבע בשיטת מייעון אוגר ישיר, ואופrnd העיד 2 בשיטת מייעון אוגר עקיף, ולכן יקודדו במילת-האוגרים יקודדו במילת-המידע נוספת אחת משותפת.</p>

מפורט הוראות המכונה:

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

קבוצת הוראות הראשונה:
אליהן הוראות הדורשות שני אופרנדים.

הוראות השיכנות לקובוצה זו הן: mov, cmp, add, sub, lea

הוראה	opcode	הפעולה המתבצעת	דוגמה	הסבר הדוגמה
mov	0	מבצעת העתקה של האופרנד (source), אופרנד המקור (הראשון) אל האופרנד השני, אופרנד היעד (destination) (בהתאם לשיטת המיעון).	mov A, r1	העתק את תוכן המשטנה A (המילה שבכתובת ז' זיכרונו) אל אוצר r1.
cmp	1	מבצעת "השוואה" בין שני האופרנדים שלה. אופן ההשוואה: תוכן אופרנד היעד (השני) מופחת מתוכן אופרנד המקור (הראשון), ללא שמירה תוצאה החישור. פועלות החישור מעודכנת דגל בשם Z ("דגל האפס") באוצר הסטטוס (PSW).	cmp A, r1	אם תוכן המשטנה A זהה לתוכנו של אוצר ז' אז דגל האפס, Z, באוצר הסטטוס (PSW) יודלק, אחרת הדגל יאפס.
add	2	אופרנד היעד (השני) מקבל את תוצאה החיבור של אופרנד המקור (הראשון) והיעד (השני).	add A, r0	אוצר 0 מקבל את תוצאה החיבור של תוכן המשטנה A ותוכנו הנוכחי של 0.
sub	3	אופרנד היעד (השני) מקבל את תוצאה החישור של אופרנד המקור (הראשון) מופרנד היעד (השני).	sub #3, r1	אוצר 1 מקבל את תוצאה החישור של הערך 3 מתוכנו הנוכחי של אוצר 1.
lea	4	lea הוא קיצור (ראשי תיבות) של load effective address זו מציבה את המعن זיכרונו המוצג על ידי התווית שבאופרנד הראשון (המקורה), אל אופרנד היעד (האופרנד השני).	lea HELLO, r1	המען שמייצגת התווית HELLO מוצב לאוצר r1.

קבוצת הוראות השנייה:

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

הוראות השיכנות לקובוצה זו הן: not, clr, inc, dec, jmp, bne, red, prn, jsr

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

הסבר הדוגמה	דוגמה	הפעולה המتبוצעת	הווראה	opcode
PC ← LINE מצביע התוכנית מקבל את המعن המוצג על ידי התווית LINE, ולפיכך הפוקודה הבאה שתבוצע תהיה במען זה.	jmp LINE	קפיצה (הסתעפות) בלתי מותנית אל ההוראה שנמצאת במען המוצג על ידי האופרנד. כלומר, כתוצאה מביצוע ההוראה, מצביע התוכנית (PC) קיבל את ערך תחיה במען זה.	jmp	9
אם ערך הדגל Z באוגר הסטטוס (PSW) (הוינו איזי : PC ← LINE)	bne LINE	bne הוא קיצור (ראשי תיבות) של branch if not equal (to zero). זה הוראות הסתעפות מותנית. מצביע התוכנית (PC) קיבל את ערך אופרנד היעד אם ערכו של הדגל Z באוגר הסטטוס (PSW) הינו 0. כזכור, הדגל Z נקבע בפקודת cmp.	bne	10
קוד ה-ascii של התו הנקרא מהקלט יכנס לאוגר r1.	red r1	קריאה שלתו מהקלט הסטנדרטי (stdin) אל האופרנד.	red	11
התו אשר קוד ה-ascii שלו נמצא באוגר r1 יודפס לפלט הסטנדרטי.	prn r1	הדפסת התו הנמצא באופרנד, אל הפלט הסטנדרטי (stdout).	prn	12
push(PC) PC ← FUNC	jsr FUNC	קריאה לשגרה (סברוטינה), מצביע התוכנית (PC) הנוכחי נדחף לתוך המחסנית שボזיכרון המחשב, והאופרנד מוכנס ל-PC.	jsr	13

קובצת ההוראות השלישי: אלו הן ההוראות ללא אופרנדים. קידוד ההוראה מורכב ממילה אחת בלבד. במקרה זה, השדות של אופרנד המקור ואופרנד היעד אינם רלוונטיים כי אין אופרנדים, וכיילו 0.

ההוראות השויות לקובצת זו הן : rts, stop

הסבר הדוגמה	דוגמה	הפעולה המتبוצעת	הווראה	opcode
PC ← pop()	rts	חרזה משיגרה. הערך שנמצא בראש המחסנית של המחשב מושך מן המחסנית, ומוכנס אל מצביע התוכנית (PC).	rts	14
התוכנית עצרת	stop	עצירת ריצת התוכנית.	stop	15

מבנה שפת האסמבלי :

שפת האסמבלי בנויה ממשפטים (statements). קובץ בשפת אסמבלי מורכב משורות המכילות ממשפטים של השפה, כאשר כלמשפט מופיע בשורה נפרדת. כלומר הינו המפרד בין משפט אחד למשפט אחר בקובץ הינו התו 'ט' (שורה חדשה).

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

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

<p>הסביר כלל</p> <p>זהו שורה המכילה אך ורק תווים לבנים (whitespace), ככלומר רק את התווים ' ' - 'ז' (רווחים וטאבים). ייתכן ובשורה אין אף תו (למעט התו ח), ככלומר השורה ריקה.</p> <p>זהו שורה בה התו הראשון הינו ' ' (נקודה פסיק). על האסמלר להתעלם לחולוטן משורה זו.</p>	<p>סוג המשפט</p> <p>משפט ריק</p>
<p>זהו משפט המנחה את האסמלר מה עליו לעשות כשהוא פועל על תכנית המקור. יש מספר סוגים של משפטי הנחיה. משפט הנחיה עשוי לגרום להקצת זיכרון ואתחול משתנים של התכנית, אך הוא אינו מייצר קידוד של הוראות מכונה המיועדות לביצוע בעת ריצת התכנית.</p> <p>זהו משפט המיין קידוד של הוראות מכונה לביצוע בלבד בצע, וטיור האופרנדים המשפט מרכיב ממש של הוראה שעל המעבד לבצע, ושל ההוראה.</p>	<p>משפט הערה</p>
<p>זהו משפט המנחה את האסמלר מה עליו לעשות כשהוא פועל על תכנית המקור. יש מספר סוגים של משפטי הנחיה. משפט הנחיה עשוי לגרום להקצת זיכרון ואתחול משתנים של התכנית, אך הוא אינו מייצר קידוד של הוראות מכונה המיועדות לביצוע בעת ריצת התכנית.</p> <p>זהו משפט המיין קידוד של הוראות מכונה לביצוע בלבד בצע, וטיור האופרנדים המשפט מרכיב ממש של הוראה שעל המעבד לבצע, ושל ההוראה.</p>	<p>משפט הנחיה</p>
<p>נת נפרט יותר לגבי סוגי המשפטים השונים.</p>	<p>משפט הוראה</p>

משפט הנחיה:

משפט הנחיה הוא בעל המבנה הבא :

בתחילת המשפט יכולה להופיע הגדרה של תווית (label). לתווית יש תחביר חוקי שיתואר בהמשך. התוויות היא אופציונאלית.

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

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

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

1. הנחיה 'data'.

פרמטרים של הנחיה 'data', הם מספרים שלמים חוקיים (אחד או יותר) המופרדים על ידי התו ' ' (פסיק). לדוגמה :

.data 7, -57, +17, 9

יש לשים לב שהפסיקים אינם חייבים להיות צמודים למספרים. בין מספר לפסק ובין פסק לבין יכולם להופיע רווחים וטאבים בכל כמות (או בכלל לא), אולם הפסיק חייב להופיע בין המספרים. כמו כן, אסור שיופיע יותר מפסק אחד בין שני מספרים, וגם לא פסק אחרי המספר האחרון או לפני המספר הראשון.

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

כלומר אם נכתב :

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

אזי יוקצו בתמונה הנתונים ארבע מילימטרים רצופות שיכילו את המספרים שמופיעים בהנחיה. התווית XYZ מזווהה עם כתובות המילה הראשונה.

אם נכתב בתכנית את ההוראה :

mov XYZ, r1

או בזמן ריצת התכנית יוכנס לאויגר r1 ערך 7.

ואילו ההוראה :

lea XYZ, r1

תכנית לאויגר r1 את ערך התווית XYZ (כלומר הכתובת בזיכרון בה מוחשן הערך 7).

2. הנקיה '.string'

הנקיה '.string' פרמטר אחד, שהוא מחרוזת חוקית. תוכי המחרוזת מקודדים לפי ערכיו ascii. המתאים, ומוכנסים אל תמונה הנתונים לפי טרם, כלתו במילה נפרדת. בסוף המחרוזת יתוסף התו '\0' (הערך המספרי 0), המסמן את סוף המחרוזת. מונה הנתונים של האסמבלי יקודם בהתאם לאויך המחרוזת (בהתאמה מקום אחד עבר התו המטיסים). אם בשורת הנקיה מוגדרת תווית, או תווית זו מקבלת את ערך מונה הנתונים (לפניהם) ומוכנסת אל טבלת הסמלים, בדומה כפי שנעשה עבור '.data'. (כלומר ערך התווית יהיה הכתובת בזיכרון שבמהלך הנקיה).

לדוגמה, הנקיה :

STR: .string "abcdef"

מקצת בתמונה הנתונים רצף של 7 מיללים, ומאתחלת את המילים לקובץ ascii של התווים לפי הסדר במחוזות, ולאחריהם הערך 0 לסיום סוף מחוזות. התווית STR מזוהה עם כתובות התחלת המחרוזת.

3. הנקיה '.entry'

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

לדוגמה, השורות :

.entry HELLO
HELLO: add #1,r1

מודיעות לאסמבלי שאפשר להתייחס מקובץ אחר לתווית HELLO המוגדרת בקובץ הנוכחי.

لتשומת לב: תווית המוגדרת בתחלת שורת entry. הינה חסרת משמעות והאסמבלי מועלם מתווית זו (אפשר שהאסמבלי יוציא הודעה אחרת).

4. הנקיה '.extern'

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

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

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

.extern HELLO

תווית או אופרנד: תווית המוגדרת בתחילת שורת `extern`. הינה חסרת משמעות והאסמבלר מעתה מונווית זו (אפשר שהאסמבלר יוציא הודעה אחרת).

משפט הוראה: משפט הוראה מורכב מהחלקים הבאים:

משפט הוראה אופציונלי.

1. שם הפעולה.
2. אופרנדים בהתאם לסוג הפעולה (בין 0 ל-2 אופרנדים).

אם מוגדרת תווית בשורת ההוראה, אז היא תוכנס אל טבלת הסמלים. ערך התווית יהיה מען הימלה הראשונה של ההוראה בתוך תMOVת הקוד שבונה האסמבלר.

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

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

есאר יש שני אופרנדים, האופרנדים מופרדים זה מזה בתו ',' (פסיק). בדומה להנחיה 'data', לא חייב להיות הצמדה של האופרנדים לפסיק או לשם הפעולה באופן בלשנו. כל כמות של ווחים ו/או ט-absטים בין האופרנדים לפסיק, או בין שם הפעולה לאופרנד הראשון, היא חוקית.

משפט הוראה עם שני אופרנדים המבנה הבא:

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

דוגמה: HELLO: add r7, B

משפט הוראה עם אופרנד אחד המבנה הבא:

label(optional): opcode target-operand

דוגמה: HELLO: bne XYZ

משפט הוראה ללא אופרנדים המבנה הבא:

label(optional): opcode

דוגמה: END: stop

ಅಧ್ಯಾತ್ಮ ಮಂಜುರಿಗಳ ಶಬದ ವರ್ಣನೆ

תווית:

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

הגדרה של תווית מסתנית בתו ': (נקודתיים).תו זה אינו מהו חלק מהתווית, אלא רק סימן המציין את סוף ההגדירה.

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

לדוגמה, התוויות המוגדרות להן הן **תוויות חוקיות**.

hEllo:

x:

He78902:

לתשומת לב: מילים שמורות של שפת האסטבלי (כלומר שם של פעולה או שם של אוגר) אין יכולות לשמש גם כשם של תווית.

התווית מקבלת את ערכה בהתאם לחברו בו היא מוגדרת. תווית המוגדרת בהנחות data. ואן תקבל את ערך מונה הנטונים (data counter) הנוכחי, בעוד שתווית המוגדרת בשורת string, תקבל את ערך מונה ההוראות (instruction counter) הנוכחי. הוראה תקבל את ערך מונה ההוראות (data counter) הנוכחי.

מספר:

מספר חוקי מתחילה בסימן אופציוני: ‘-’, ‘+’, ‘או’ ‘+’, ולאחריו סדרה של ספרות בסיס עשרוני. לדוגמה: 5,76, +123 הם מספרים חוקיים. אין תמייה בשפת אסטבלי ביצוג בסיס אחר מאשר עשרוני, ואין תמייה במספרים שאינם שלמים.

מחרוזת:

מחרוזת חוקית היא סדרת תוו ascii נוראים (שניתנים לחדפסה), המוקפים במרקאות כפולות (המרקאות אין נחשבות חלק מהמחרוזת). דוגמה למחרוזת חוקית: “hello world”.

פקוד השדה A,R,E בקוד המכונה

בכל מילה בקוד המכונה של הוראה (לא של נתוניים), האסטבלי מכניס מידע עבור תהליך הקישור והטיעינה. זהו השדה A,R,E (שלוש הסיביות הימניות 0,1,0 בהתאמה). המידע ישמש לתיקונים בקוד בכל פעם שייטען לזרוך הרצה. האסטבלי בונה מלכתחילה קוד שמיועד לטיענה החל מכתובת 100. התקיונים יאפשרו לטען את הקוד בכל פעם למקומות אחרים, בלי צורך לחזור על תהליך האסטבלי.

בכל מילה של הוראה, בדיק אחת משלש הסיביות של השדה E,R,A מכילה 1, ושתי הסיביות האחרות מאופסות. מפרט שיטות המיעון שהוצג קודם מציין כי שיטה 1 בכל שיטת מעון.

סיבית ‘A’ (קייזר של Absolute) בא להציג שתוכן המילה איינו תלוי במקום בזיכרון בו ייטע בפועל קוד המכונה של התכנית בעת ביצועה (למשל מילה המכילה אופrnd מיידי).

סיבית ‘R’ (קייזר של Relocatable) בא להציג שתוכן המילה תלוי במקום בזיכרון בו ייטע בפועל קוד המכונה של התכנית בעת ביצועה (למשל מילה המכילה כתובת של תווית המוגדרת בקובץ המקור הנוכחי).

סיבית ‘E’ (קייזר של External) בא להציג שתוכן המילה תלוי בערכו של סמל חיצוני (External) (למשל מילה המכילה כתובת של תווית שמוגדרת בקובץ מקור אחר).

אסטבלי עם שני מעברים

כאשר מקבל האסטבלי תכנית בשפת אסטבלי, עליו לעבור על התכנית פעממים. במעבר הראשון, יש לzechot את הסמלים (תוויות) המופיעים בתוכנית, ולתת לכל סמל ערך מספרי שהוא המען בזיכרון שהסמל מייצג. במעבר השני, באמצעות ערכי הסמלים, וכן קוד-הפעלה ומספרי האוגרים, בונים את קוד המכונה.

לוגינה: האסמבילר מקבל את התוכנית הבאה בשפת אסמבילר:

```

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

```

קוד המכוונה של התוכנית (הוראות ונתונים) נבנה כך שיתאים לטעינה בזיכרון החל מען 100 (עשרוני).
התרגום של תוכנית הדוגמה לקוד בינארי מוצג להלן:

Decimal Address	Source Code	Explanation	Binary Machine Code
0100	MAIN: add r3, LIST	First word of instruction	001010000010100
0101		Source register 3	000000011000100
0102		Address of label LIST	000010000010010
0103	LOOP: prn #48	Immediate value 48	110000000000100
0104			0000000110000100
0105	lea STR, r6	Address of label STR	010000101000100
0106		Target register 6	000000111101010
0107			000000000010100
0108	inc r6	Target register 6	011100001000100
0109			000000000010100
0110	mov *r6,K	Source register 6	000001000010100
0111		Address of label K	000000000010000
0112			000010000010100
0113	sub r1, r4	Source register 1 and target register 4	001110001000100
0114			000000000011000
0115	cmp r3, #-6	Source register 3	0001100000001100
0116		Immediate value -6	000000000011000
0117			111111110101000
0118	bne END	Address of label END	1010000000010100
0119			0000011110010100
0120	dec K	Address of label K	0111000000010100
0121			0000100000010100
0122	jmp LOOP	Address of label LOOP	1001000000010100
0123			0000011001101010
0124	END: stop	Address of label END	1111000000000100
0125	STR: .string "abcd"	Ascii code 'a'	0000000000000000
0126		Ascii code 'b'	0000000000000000
0127		Ascii code 'c'	0000000000000000
0128		Ascii code 'd'	0000000000000000
0129		Ascii code '\0' (end of string)	0000000000000000
0130	LIST: .data 6, -9	Integer 6	0000000000000000
0131		Integer -9	1111111111010111
0132	.data -100	Integer -100	1111111100111010
0133	K: .data 31	Integer 31	0000000000011111

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

אפשר פשוט לעיין בטבלה ולמצוא את הקוד הבינארי השקול. כדי לעשות המרה לבינארי של אופרנדים שהם מענים סמליים (תוויות), יש צורך לבנות טבלה דומה. אולם בהבדל מהקודים של הפעולות, הידיעים מראש, הרו המענים בזיכרון עבור הסמלים בשימוש התכנית אינם ידועים, עד אשר תוכנית המקור נסרקה כולה ונתגלו כל הגדרות הסמלים.

למשל, בקוד לעיל, האסטמבלר אינו יכול לדעת שהסמל END אמרור להיות משוויך לערך 124 (עשרוני), והסמל K אמרור להיות משוויך לערך 133, אלא רק לאחר שנקראו כל שורות התכנית.

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

במעבר הראשון נבנית טבלה סמלים בזיכרון, ובה לכל סמל שבתוכנית המקור משוויך ערך מסוים, שהוא מען בזיכרון. בדוגמה לעיל, טבלה הסמלים לאחר מעבר ראשון היא:

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

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

لتשומת לב: תפקיד האסטמבלר, על שני המעברים שלו, לתרגם קובץ מקור לקוד בשפת מכונה. בגמר פועלות האסטמבלר, התכנית טרם מוכנה לטעינה לזכרון לצורך ביצוע. קוד המכונה חייב לעבור לשלי הקיישור/טעינה, ורק לאחר מכן לשלב הביצוע (שלבים אלה אינם חלק מהממש).

המעבר הראשון

במעבר הראשון נדרשים כלים כדי לקבוע איזה מען ישוויך לכל סמל. העיקרונו הבסיסי הוא לספור את המיקומות בזיכרון, אותן תפיסות ההוראות. אם כל הוראה תיתען בזיכרון למקום העוקב להוראה הקודמת, תציין ספרה כזאת את מען הוראה הבאה. הספרה נעשית על ידי האסטמבלר ומוחזקת במונה ההוראות (IC). ערכו ההתחלתי של IC הוא 100 (עשרוני), וכך קוד המכונה של הוראה הראשונה נבנה כך שייתען לזכרון לחיל מען 100. IC מעתדען בכל שורת הוראה המקצת מקום בזיכרון. לאחר שהאסטמבלר קובע מהו אורק הוראה, IC מוגדל במספר התאים (מילימ) הנוטפסים 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, וכל אופרנד בקידוד מתאים. אך פעולת החלפה אינה כה פשוטה. ההוראות משתמשות בשיטות מייעון מגוונות לאופרנדים. אותה פעולה יכולה לקבל משמעויות שונות, בכל אחת משיטות המייעון, ולכן תואמו לה קידודים שונים לפי שיטות המייעון. לדוגמה, פעלת ההזזה sow יכולה להתיחס להעתקת תוכן תא זיכרון לאונר, או להעתקת תוכן אונר אחר, וכן הלאה. לכל אפשרות כזו של sow עשוי להתאים קידוד שונה.

על האסטמבלר לסרוק את שורת ההוראה בשלה, ולהחליט לגבי הקידוד לפי האופרנדים. בדור כל מתחלק הקידוד לשדה של שם הפעולה, ושדות נוספים המכילים מידע לגבי שיטות המייעון.

במודל המחשב שלנו קיימת גמישות לגבי שיטת המיעון של כל אחד מהאופרנדים בנפרד. העריך: זה לא בהכרח כך בכל מחשב. יש מחשבים בהם, למשל, כל הפקודות הן בעלות אופרנד אחד (והפעולות מתבצעות על אופרנד זה ועל אוגר קבוע). יש גם מחשבים עם פקודות של שלשה אופרנדים (כאשר האופרנד השלישי משמש לאחסון תוצאה הפעולה), ועוד אפשרויות אחרות.

כאשר נתקל האסטմבלר בתווית המופיעיה בתחילת השורה, הוא יודע שלפנוי הגדרה של תווית, והוא משייך לה מען – תוכנו הנוכחי של ה-IC. כך מקבלות כל התוויות את מענהן בעת ההגדירה. תוויות אלה מוכנסות לטבלת הסמלים, המכילה בנוסף לשם התווית גם את המען ומאפיינים נוספים. כאשר תהיה התיחסות לתווית באופרנד של הוראה כלשהי, יוכל האסטמבלר לשולף את המען המתאים מטבלת הסמלים.

הוראה יכולה להתיחס גם לסדר טריטם הוגדר עד כה בתכנית, אלא יוגדר רק בהמשך התכנית. להלן דוגמה, הוראות הסתעפות מען שמוגדר עלי ידי התווית A שמופיעיה רק בהמשך הקוד:

bne A
•
•
•
A:

כאשר מגיע האסטמבלר לשורת ההסתעפות (A bne), הוא טרם נתקל בהגדרת התווית A וכמוון לא נתן לה מען, ולכן אינו יכול להחליף את הסמל A (האופרנד של ההוראה bne) בمعונו בזיכרון. נואה בהמשך כיצד נפתרת בעיה זו.

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

המעבר השני

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

לשם כך עובר האסטמבלר שנית על כל קובץ המקור, וمعدכן את קוד המcona של האופרנדים המשמשים בסמלים, באמצעות ערכיו בטבלת הסמלים. זהו המעבר השני, ובסיומו תהיה התוכנית מתרגמת בשלמותה לקוד מכונה.

הפרדת הוראות ונתונים

בתכנית מבחינים בשני סוגי של תוכן: הוראות ונתונים. יש לארגן את קוד המcona כך שתתהייה הפרדה בין הנתונים וההוראות. הפרדת ההוראות והנתונים לקטעים שונים בזיכרון היא שיטה עדיפה על פני הצמדת הגדירות הנתונים להוראות המשמשות בהן.

אחד הסכנות הטමונות באյ הפרדת ההוראות מהנתונים היא, שלפעמים עלול המעבד, בעקבות שגיאה לוגית בתכנית, לנסה "לבצע" את הנתונים כאילו היו הוראות חוקיות. למשל, שגיאה שיכולה לגרום לתופעה כזו הסתעפות לא נכונה. התכנית מבוסנת לא לעבוד נכון, אך לרוב הנזק הוא יותר חמוץ, כי נוצרת חריגת חומרה ברגע שהמעבד מבצע פעולה שאינה חוקית.

האסטמבלר שלנו חייב להפריד, בקוד המcona שהוא מיצר, בין קטוע הנתונים לקטוע ההוראות. ולומר בקובץ הפלט (בקוד המcona) תהיה הפרדה של הוראות ונתונים לשני קטיעים נפרדים, ואילו בקובץ הקלט אין חובה שתהייה הפרדה כזו. בהמשך מתואר אלגוריתם של האסטמבלר, ובו פרטים כיצד לבצע את הפרדה.

גילוי שגיאות בתכנית המקור

האסטמבלר אמרור לגנות ולדוח על שגיאות בתחביר של תכנית המקור, כגון פעולה שאינה קיימת, מספר אופרנדים שני, סוג אופרנד שאינו מתאים לפועלה, שם אוגר לא קיים, ועוד שגיאות אחרות. כמו כן מודיא האסטמבלר שככל סמל מוגדר פעם אחת בדיק.

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

האסטמבלר ידפיס את הודעת השגיאה אל הפלט הסטנדרטי `stdout`. בכל הודעת שגיאה יש לעזין גם את מסטר השורה בקובץ המקור בה זהותה השגיאה.

לתשומת לב: האסטמבלר איןנו עוצר את פעולתו אחרי שנמצאה השגיאה הראשונה, אלא ממשיך לעבור על הקלט כדי לגנות שגיאות נוספות, ככל שישן. כМОון שאין כל טעם לייצר את קבצי הפלט אם נתגלו שגיאות (ممילא אי אפשר להשלים את קוד המוכנה).

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

שיטות מייעון חוקיות עבור אופרנד היעד	שיטות מייעון חוקיות עבור אופרנד המקור	שם ההוראה	opcode
1,2,3	0,1,2,3	mov	0
0,1,2,3	0,1,2,3	cmp	1
1,2,3	0,1,2,3	add	2
1,2,3	0,1,2,3	sub	3
1,2,3	1	lea	4
1,2,3	אין אופרנד מקור	clr	5
1,2,3	אין אופרנד מקור	not	6
1,2,3	אין אופרנד מקור	inc	7
1,2,3	אין אופרנד מקור	dec	8
1,2	אין אופרנד מקור	jmp	9
1,2	אין אופרנד מקור	bne	10
1,2,3	אין אופרנד מקור	red	11
0,1,2,3	אין אופרנד מקור	prm	12
1,2	אין אופרנד מקור	jsr	13
אין אופרנד יעד	אין אופרנד מקור	rts	14
אין אופרנד יעד	אין אופרנד מקור	stop	15

תහיליך העבודה של האסטמבלר

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

האסטמבלר מתחזק שני מערכיים, שייקראו להן תמונה ההוראות (code) ותמונה הנתונים (data). מערכיים אלו נתונים למשה תמונה של זיכרון המכונה (כל איבר במערך הוא בגודל מילה של המכונה, כולם 15 סיביות). במערך ההוראות בונה האסטמבלר את הקידוד של ההוראות המכונה שנקראו במהלך המעבר על קובץ המקור. במערך הנתונים מכניס האסטמבלר את קידוד הנתונים שנקראו מקובץ המקור (שורות הנחיה מסווג 'data' ו-'string').

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

בנוסף, מתחזק האסטמבלר טבלה, אשר בה נאספנות כל התוויות בהן נתקל האסטמבלר במהלך הפעלה על קובץ המקור. לטבלה זו קוראים טבלת-הסמלים (symbol-table). בכל סמל נשמרם המעבר שמו הסמל, ערכו המספרי, ומאפיינים שונים, כגון המיקום (data או code), וסוג הסמל בטבלה שמו entry או external.

3. שורת הוראה או שורת הערה: האסטמבלר בונה את טבלת הסמלים ואת השילד של תМОונת הזיכרון (הוראות ונתונים) בהתאם למקורו.

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

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

5. שורת הוראה: האסטמבלר מנתה את השורה ומפענח מהי ההוראה, ומהן שיטות המיעון של האופרנדים. מספר האופרנדים נקבע בהתאם להוראה שנמצאה. שיטות המיעון נקבעות בהתאם לתחביר של כל אופרנד, כפי שהסביר לעיל במפרט שיטות המיעון. למשל, התו '#' מציין מיון מיידי, תווית מצינית מיון ישר, שם של אוגר מציין מיון אוגר ישיר, ועוד'.

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

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

כעת האסטמבלר קובע לכל אופרנד את ערכו באופן הבא:

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

האסטמבלר מכניס למערך ההוראות, בכניסה אליה מציביע מונה ההוראות IC, את קוד המילה הראשונה של ההוראה (בפורמט קידוד כפי שתואר קודם). מילה זו מכילה את קוד הפעולה, ואת מספרי שיטות המיון. ה- IC מקודם ב-1.

זכור שכאשר יש רק אופרנד אחד (כלומר אין אופרנד מקור), הסיביות של שיטת המיון של אופרנד המקור יכלו תמיד 0. בדומה, אם זהה ההוראה ללא אופרנדים (stop, ts), אז הסיביות של שיטות המיון של שני האופרנדים יכלו 0.

6. שורת הנחיה: אם זהה ההוראה בעלת אופרנדים (אחד או שניים), האסטמבלר "משרין" מקום במערך ההוראות עבור מילות-המידע הנוספות הנדרשות בהוראה זו, ומקדם את IC בהתאם. כאשר אחד או שני האופרנדים הם בשיטת מיון אוגר-ישיר, אוגר-עקבף, או מיידי, האסטמבלר מקודד גם את המילים הנוספות הרלוונטיות במערך ההוראות. מילת-המידע של שיטת מיון ישר תישאר ללא קידוד.

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

I. '.data' – האסטמבלר קורא את רשימת המספרים, המופיעה לאחר '.data', מכניס כל מספר אל מערך הנתונים, ומקדם את מצביע הנתונים DC ב-1 עבר כל מספר שהוכנס.

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

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

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

III. '.entry' – זהה להנחה לאסטמבלר לאפיין את התווית הנתונה כאופrndכ-c-entry בטבלת הסמלים. בעת הפיקט קבצי הפלט (ראו בהמשך), התווית תירשם בקובץ ה-entries. לתשומת לב: זה לא נחשב כסוגיה אם בקובץ המקור מופיע יותר מהנחיית entry. אחת עם אותה תווית כאופrnd. המופעים הנוספים אינם מושפעים דבר, אך גם אינם מפריעים.

IV. '.extern' – זהה הערה על סמל (תווית) המוגדר בקובץ מקור אחר, והקובץ הנוכחי עושים בו שימוש. האסטמבלר מכניס את הסמל המופיע כאופrnd לטבלת הסמלים, עם הערך 0 (הערך האמתי לא ידוע, ויקבע רק בשלב הקישור), ועם המאפיין external. לא ידוע באיזה קובץ נמצאת הגדרה הסמל, ואין זה רלוונטי עבור האסטמבלר. לתשומת לב: זה לא נחשב כסוגיה אם בקובץ המקור מופיע יותר מהנחיית extern. אחת עם אותה תווית כאופrnd. המופעים הנוספים אינם מושפעים דבר, אך גם אינם מפריעים.

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

בסוף המעבר הראשון, האסטמבלר מעדכן בטבלת הסמלים כל סמל המופיע כ-.data, על ידי הוספת (100) + IC (עשרות) לערכו של הסמל. הסיבה לכך היא שבתמונה הכלולת של קוד המכינה, תMOVNT_R נתוניים מופרדים מתמונה הוראות, וכל הנתוניים נדרשים להופיע בקוד המכינה אחרי כל ההוראות. סמל מסווג data הוא תווית בתמונה הנתונים, והעדכו מוסיף לערך הסמל (כלומר לכתובתו בזיכרונו) את האורך הכלול של תמונה ההוראות, בתוספת כתובת התחלה הטעינה של הקוד, שהיא 100.

טבלת הסמלים מכילה כעת את כל הערכים הנוחוצים להשלמת תMOVNT_R (למעט ערכים של סמלים חיצוניים).

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

אלגוריתם שלוי של האסטמבלר

לחידוד ההבנה של תהליך העבודה של האסטמבלר, נציג להלן אלגוריתם שלו למעבר הראשון ולמעבר השני.

لتשומת לב: אין חובה להשתמש דווקא באלגוריתם זה.

כאמור, אנו מחלקים את תMOVNT_R המכונה לשני חלקים: תMOVNT_R (code), ותMOVNT_R (data). לכל חלק נתזקק מונה נפרד: IC (מונה ההוראות) ו-DC (מונה הנתונים).

כמו כן, נסמן ב- L את המספר הכלול של מיללים שתופס קוד המכונה של הוראה נתונה.
נגנה את קוד המכונה כך שיתאים לטעינה לזיברונו החל מכתובת 100.
בכל מעבר מתחילה לקרוא את קובץ המקור מההתחלת.

מעבר ראשון

1. אתחל $0 \leftarrow IC$.
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 הנחיה לאחסון נתונים, למשל, הכנס האמצעי `data`. או `string`. אם לא, עברו ל-8.
6. אם יש הגדרת סמל (תוויות), הכנס אותו לטבלת הסמלים עם המאפיין `data`. ערך הסמל יהיה `DC`. אם הסמל כבר נמצא בטבלה יש להודיע על שגיאה.
7. זהה את סוג הנתונים, קודן אותם בתמונה הנתונים, ועדכן את מונה הנתונים `DC` בהתאם לאורך הנתונים. חזור ל-2.
8. אם זהה הנחיה `extern`. או `entry`. ? אם לא, עברו ל-11.
9. אם זהה הנחיה `entry`. חזור ל-2 (ההנחהتطופל במעבר השני).
10. אם זו הנחיה `extern`, הכנס כל סמל (אחד או יותר) המופיע כאופרנד של ההנחה לתוך טבלת הסמלים, ללא ערך, עם המאפיין `external`. חזור ל-2.
11. זוהי שורת הוראה. אם יש הגדרת סמל, הכנס אותו לטבלת הסמלים עם המאפיין `code`.
12. ערכו של הסמל יהיה $IC + 100$ (אם הסמל כבר נמצא בטבלה יש להודיע על שגיאה).
13. חפש את שם הפעולה בטבלת שמות הפעולות, ואם לא נמצא, אז הודיע על שגיאה בשם ההוראה.
14. נתח את מבנה האופרנדים של ההוראה וחשב את L (מספר המיללים שתופסת ההוראה בקוד הבינארי).
15. עדכן $L \leftarrow IC + L$, וחזור ל-2.
16. קובץ המקור נקרא בשלמותו. אם נמצא שגיאות במעבר הראשון, עצור כאן.
17. עדכן בטבלת הסמלים את ערכו של כל סמל המופיע כ- `data`, `string`, `extern` והוספה $IC + 100$ (ראה הסבר בהמשך).
18. התחל מעבר שני.

מעבר שני

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

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

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

נבע מעבר ראשון על הקוד לעיל, ובנה את טבלת הסמלים. כמו כן, נשלים במעבר זה את הקידוד של תМОנות הנתונים, ושל המילה הראשונה של כל הוראה. כמו כן, נקודד מילוט-מידע ומספרות של הוראה, ככל שקידוד זה אינו תלוי בערך של סמל. את מילות-המידע שעדיין לא ניתן לקודד במעבר הראשון נסמן ב"!!!" בדוגמה להלן.

Decimal Address	Source Code	Explanation	Binary Machine Code
0100	MAIN: add r3, LIST	First word of instruction Source register 3 Address of label LIST	001010000010100 000000011000100 ?
0101			
0102			
0103	LOOP: prn #48	Immediate value 48	1100000000000100 000000110000100
0104			
0105	lea STR, r6	Address of label STR Target register 6	010000101000100 ?
0106			
0107			
0108	inc r6	Target register 6	011100001000100 000000000110100
0109			
0110	mov *r6,K	Source register 6 Address of label K	000001000010100 000000110000100 ?
0111			
0112			
0113	sub r1, r4	Source register 1 and target register 4	00110001000100 000000001100100
0114			
0115	cmp r3, #-6	Source register 3 Immediate value -6	0001100000001100 000000011000100 11111111010100
0116			
0117			
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	1001000000010100 ?
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)	0000000000000000
0130	LIST: .data 6, -9	Integer 6	00000000000000110
0131		Integer -9	1111111111101111
0132	.data -100	Integer -100	111111110011100
0133	K: .data 31	Integer 31	00000000000011111

טבלת הסמלים היא:

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

נבע עתה את המעבר השני. נשלים באמצעות טבלת הסמלים את הקידוד החסר במילims המסומנים ???.
הקוד הבינארי בצוותו הסופית akan זהה לקוד שהוצע בתחלת הנושא "אסמבLER עם שני מעבריים".

הערה: כאמור, האסמבLER בונה קוד מכונה כך שיתאים לטעינה לזכרון החל מכתובת 100 (עשרוני).
אם הטעינה בפועל (לצורך הרצת התכנית) תהיה לכתובת אחרת, יידרשו תיקונים בקוד הבינארי
בשלב הטעינה, שיוכנסו בעזרת מידע נוסף שהאסמבLER מכין בקבצי הפלט (ראו בהמשך).

Decimal Address	Source Code	Explanation	Binary Machine Code
0100	MAIN: add r3, LIST	First word of instruction Source register 3 Address of label LIST	001010000010100 000000011000100 000010000010010
0101			
0102			
0103	LOOP: prn #48	Immediate value 48	110000000000100 000000110000100
0104			
0105	lea STR, r6	Address of label STR Target register 6	010000101000100 000001111101010 000000000110100
0106			
0107			
0108	inc r6	Target register 6	011100001000100 000000000110100
0109			
0110	mov *r6,K	Source register 6 Address of label K	000001000010100 000000110000100 000010000101010
0111			
0112			
0113	sub r1, r4	Source register 1 and target register 4	001110001000100 000000001100100
0114			
0115	cmp r3, #-6	Source register 3 Immediate value -6	000110000001100 000000011000100 11111111010100
0116			
0117			
0118	bne END	Address of label END	101000000010100 000001111001010
0119			
0120	dec K	Address of label K	011100000010100 000010000001010
0121			
0122	jmp LOOP	Address of label LOOP	100100000010100 000001100111010
0123			
0124	END: stop		111100000000100
0125	STR: .string "abcd"	Ascii code 'a'	000000001100001
0126		Ascii code 'b'	000000001100010
0127		Ascii code 'c'	000000001100011
0128		Ascii code 'd'	000000001100100
0129		Ascii code '\0' (end of string)	000000000000000
0130	LIST: .data 6, -9	Integer 6 Integer -9	000000000000110 11111111110111
0131			
0132	.data -100	Integer -100	111111110011100
0133	K: .data 31	Integer 31	000000000011111

בסוף המעבר השני, אם לא נתגלו שגיאות, האסטמבלר 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 האסטמבלר

בפעולת של האסטמבלר, יש להעביר אליו באמצעות ארגומנטים של שורת הפקודה (command line arguments) רשימה של שמות קבצי מקור (אחד או יותר). אלו הם קבצי טקסט, ובינם תכניות בתחריר של שפת האסטמבל שהוגדרה בມינ' זה.

האסטמבלר פועל על כל קובץ מקור בנפרד, ויצרת עבורו קבצי פלט כדלקמן:

- קובץ object, המכיל את קוד המוכנה.
- קובץ externals, ובו פרטיטים על כל המיקומות (הכתובות) בקוד המוכנה בהם מקודד ערך של סמל חזוני (סמל שהוגדר באמצעות ההנחיה extern, ומופיע בטבלת הסמלים כ-external).
- קובץ entries, ובו פרטיטים על כל סמל שימושר ננקודת כניסה (סמל שהופיע כאופרנד של הנחיתת entry, ומופיע בטבלת הסמלים כ-entry).

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

שמות קבצי המקור חיברים להוות עם הסיומת ".as". למשל, השמות x.as, y.as, ו-as.htm שמות חוקיים. העברת שמות הקבצים הללו כארוגומנטים לאסטמבלר נעשית לא ציון הסיומת.

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

assembler x y hello

תירץ את האסטמבלר על הקבצים:x.as, y.as, hello.as :

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

לדוגמה, בהפעלת האסטמבלר באמצעות שורת הפקודה : assembler x יוצר קובץ פלט obj.x, וכן קבצי פלט x.ent ו- x.ext. ככל שיש הנחיתות entry, או extern. בקובץ המקור. נציג כתע את הפורמטים של קבצי הפלט. דוגמאות יובאו בהמשך.

פורמט קובץ ה- object

קובץ זה מכיל את תМОונת הזיכרון של קוד המוכנה, בשני חלקים: תМОונת ההוראות ראשונה, ואחריה ובצמוד תМОונת הנתונים.

כזכור, האסטמבלר מקודד את ההוראות כך שתМОונת ההוראות תתאים לטיענה החל מכתובת 100 (עשרוני) בזיכרון. נשים לב שرك בסוף המעבר הראשוני יודעים מהו הגודל הכלול של תМОונת ההוראות. מכיוון שתМОונת הנתונים נמצאת אחרי תМОונת ההוראות, גודל תМОונת ההוראות משפיע על הכתובות בתМОונת הנתונים. זו הסיבה שבגללה היה צריך לעדכן בטבלת הסמלים, בסוף המעבר הראשוני, את ערכי הסמלים המופיעים כ-data (כזכור, הוסףנו לכל סמל כזה את הערך 100 C+II). במעבר השני, השלים את הקידוד משתמש בערכים המעודכנים של הסמלים, המותאמים למבנה המלא והסופי של תМОונת הזיכרון.

כעת האסמבולר יכול לכתוב את תמונה הזיכרון בשלמותה בתוך קובץ פלט (קובץ ה-object).

השורה הראשונה בקובץ ה-object היא "cotract", המכילה שני מספרים (בסיס עשרוני): הראISON הוא האורך הכלול של תמונה החרואות (במילוט זיכרון), והשני הוא האורך הכלול של תמונה הנתונים (במילוט זיכרון). בין שני המספרים יש רווח אחד.

השורות הבאות בקובץ מכילות את תמונה הזיכרון. בכל שורה שני שדות: כתובות של מילה בזיכרון, ותוכן המילה. הכתובת תירשם בסיס עשרוני באربע ספרות (כולל אפסים מובילים). תוכן המילה יירשם בסיס אוקטלי ב-5 ספרות. בין שני השדות יש רווח אחד.

פורמט קובץ ה-entries

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

פורמט קובץ ה-externals

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

כמו כן ישיתכן ויש מספר כתובות בקוד המכונה בהן הקידוד מתאפיין לאותו סמל חיצוני. לכל כתובת צו זה יהיה שורה נפרדת בקובץ ה-externals.

נדגים את הפלט שמייצר האסמבולר עבור קובץ מקור בשם ps.as הנלוון להלן.

```
; file ps.as

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

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

Decimal Address	Source Code	Explanation	Binary Machine Code
0100	MAIN: add r3, LIST	First word of instruction Source register 3 Address of label LIST	001010000010100 000000011000100 000010001001010
0101			
0102			
0103	jsr fn1	Address of label fn1 (external)	110100000010100 00000000000000001
0104			
0105	LOOP: prn #48	Immediate value 48	1100000000001100 000000110000100
0106			
0107	lea STR, r6	Address of label STR Target register 6	010000101000100 000010000100010 000000000110100
0108			
0109			
0110	inc r6	Target register 6	011100001000100 0000000000110100
0111			
0112	mov *r6, L3	Source register 6 Address of label L3 (external)	000001000010100 000000110000100 0000000000000001
0113			
0114			
0115	sub r1, r4	Source register 1 and target register 4	001110001000100 0000000001100100
0116			
0117	cmp r3, #-6	Source register 3 Immediate value -6	000110000001100 0000000011000100 111111111010100
0118			
0119			
0120	bne END	Address of label END	101000000010100 000010000011010
0121			
0122	add r7, *r6	Source register r0 and target register 6	0010100000100100 000000111110100
0123			
0124	clr K	Address of label K	010100000010100 000010001100010
0125			
0126	sub L3, L3	Address of label L3 (external) Address of label L3 (external)	001100100010100 0000000000000001 0000000000000001
0127			
0128			
0129	jmp LOOP	Address of label LOOP	100100000010100 000001101001010
0130			
0131	END: stop	Ascii code 'a'	1111000000000100 000000001100001
0132	STR: .string "abcd"	Ascii code 'b'	000000001100010
0133		Ascii code 'c'	000000001100011
0134		Ascii code 'd'	000000001100100
0135		Ascii code '\0' (end of string)	0000000000000000
0136		Integer 6	0000000000000110
0137	LIST: .data 6, -9	Integer -9	11111111110111
0138		Integer -100	111111110011100
0139	.data -100	Integer 31	0000000000011111
0140	K: .data 31		

פלט תוכן קבוע הפלט של הזוגמה.
היקום psd:

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.ext

:ps.ent

MAIN 100
LIST 137

fn1 0104
L3 0114
L3 0127
L3 0128

סיכום והנחיות כלליות

• גודל תוכנית המקור ניתנת ככלט לאסטמבלר אינו ידוע מראש, ולכן גם גודלו של קוד המוכנה אינו צפוי מראש. אולם בצדיה להקל בימוש האסטמבלר, מותר להניח גודל מקסימלי. לפיכך יש אפשרות להשתמש במערכות לאחסן תמונות קוד המוכנה בלבד. כל מבנה נתונים אחר (למשל טבלת הסמלים), יש למשול באופן ייעיל וחסכוני (למשל באמצעות רשימה מקושרת והקצתה זיכרון דינמי).

• השמות של קבצי הפלט צריכים להיות תואמים לשם קובץ הקלט, למעט הסיומות. למשל, אם קובץ הקלט הוא `prog.as` אז קבצי הפלט שייצרו הם: `prog.ob`, `prog.ext`, `prog.ent`.

• מתכונת הפעלת האסטמבלר צריכה להיות כפי הנדרש בממ"ן, ללא שינוים כלשהם. כמובן, ממשך המשתמש יהיה אך ורק באמצעות שורת הפקודה. בפרט, שמות קבצי המקור יועברו לתכנית האסטמבלר כארגומנטים בשורת הפקודה. אין להוסיף תפריטי קלט אינטראקטיביים, חלונות גרפיים למיניהם, וכו'.

• יש להקפיד לחלק את מימוש האסטמבלר למספר מודולים (קבצים בשפת C) לפי משימות. אין לרכז שימושים מסוימים 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 המיעו החוקיות לכל פעולה, וכו').

• יש להקפיד ולתעד את המימוש באופן מלא וברור, באמצעות הערות מפורטות בקוד.

• יש לאפשר תווים לבנים עודפים בקובץ הקלט בשפת אסטמבלר. למשל, אם בשורת הוראה יש שני אופרנדים המופרדים בפסיק, או לפניו ואחריו הפסיק מותר להיות שלו רוחניים וטאבבים יתעלם מתווים לבנים מיותרים (כלומר יידלג עליהם).

• הקלט (קוד האסטמבלר) 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 למסך הודעות מפורטות לכל הniton, כדי שאפשר יהיה להבין מה והיכן כל שגיאה. כמובן שם קובץ קלט מכיל שגיאות, אין טעם להפיק עבورو את קבצי הפלט (`ent`, `ob`, `ext`).

• **תס ונסלים פרק ההסבירים והגדרת הפרויקט.**

בשאלות ניתן לנפות לקבוצת הדיוון באתר הקורס, ועל כל אחד מהמנחים בשעות הקבלה שלהם.

להזיכרים, באפשרותו של כל סטודנט לנפות לכל מנהה, לאו דווקא למנהל הקבוצה שלו, לקבלת עוזה. שוב מומלץ לכל אלה שטרם בדקו את התכנים באתר הקורס לעשות זאת. נשאלות באתר זה הרבה שאלות בנושא חומר הלימוד והממי"נים, והתשובות יכולות להועיל להם.

לתשומתיכם: לא ניתן דחיה בהגשת הממי"ן, פרט לנסיבות מיוחדות כגון מילואים או מחלת ממושכת. במקרים אלו יש לבקש ולקבל אישור מראש מוצאות הקורס.

בהצלחה!