האוניברסיטה הפתוחה

המחלקה למתמטיקה ולמדעי המחשב

מסמך אפיון ותכנון פרויקט

פרויקט גמר – כתיבת אסמבלר בשפת C

מעבדה בתכנות מערכות

מס' קורס 20465

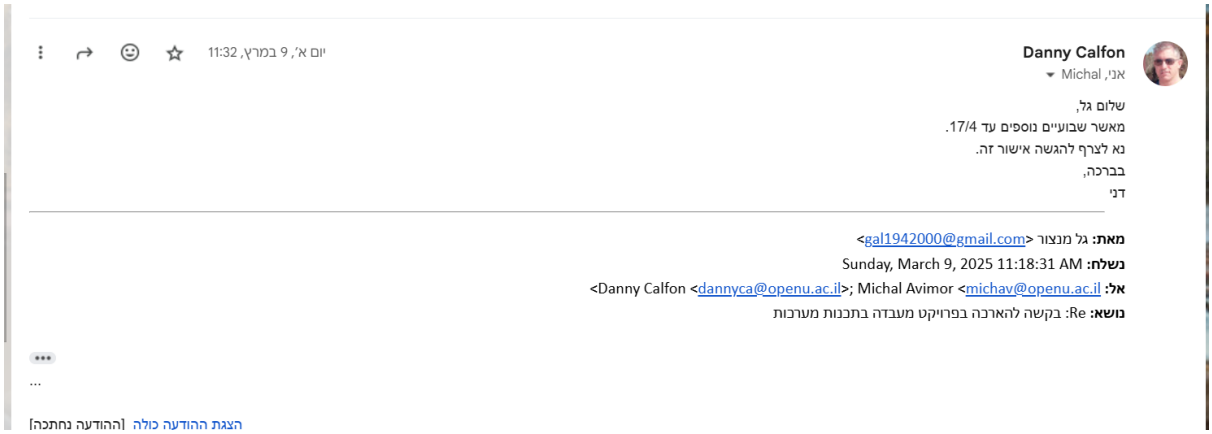
מגישת הפרויקט: גל מנצור

ת"ז: 206664369

סמסטר: 2025א'

תאריך הגשה: 14.4.2025 (ניתנה הארכת זמן ללא קנס)

קבלת הארכת זמן – ללא קנס – מהמנחה דני כלפון עד ה-17.4:



Danny Calfon
אני, Michal
יום א', 9 במרץ, 11:32

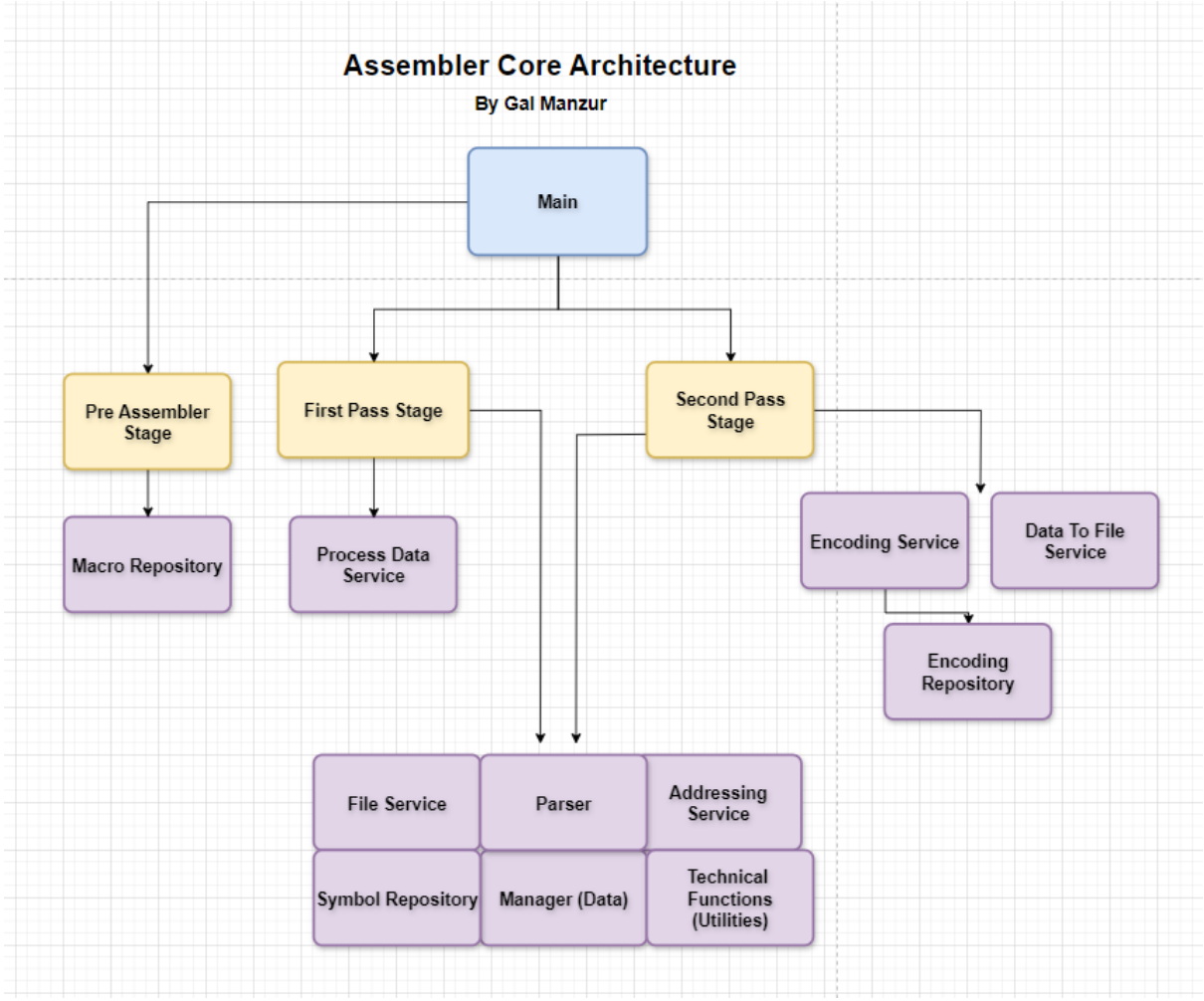
שלום גל,
מאשר שבועיים נוספים עד 17/4.
נא לצרף להגשה אישור זה.
בברכה,
דני

מאת: גל מנצור <gal1942000@gmail.com>
נשלח: Sunday, March 9, 2025 11:18:31 AM
אל: Danny Calfon <dannyca@openu.ac.il>; Michal Avimor <michav@openu.ac.il>
נושא: Re: בקשה להארכה בפרויקט מעבדה בתכנות מערכות

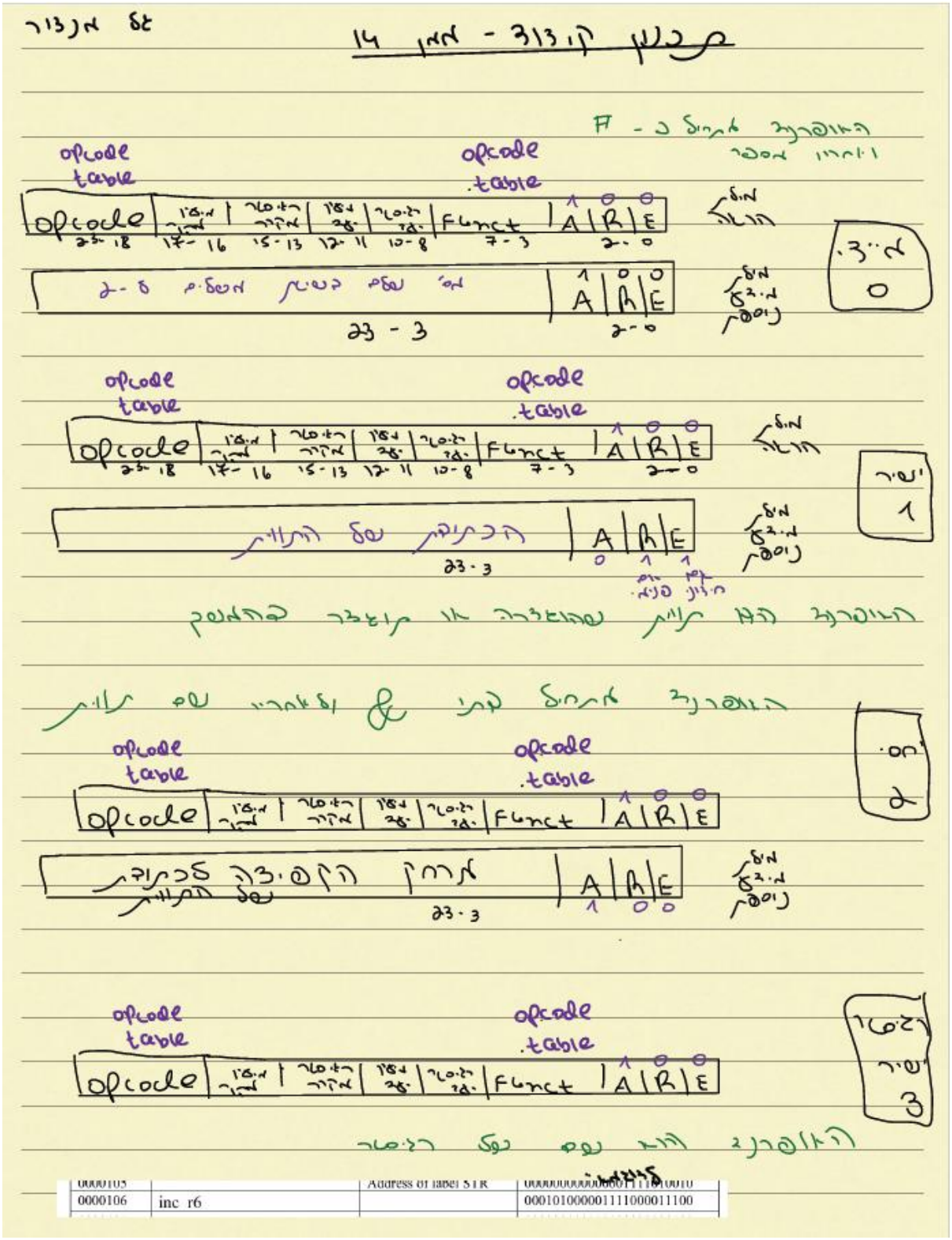
...

הצגת ההודעה כולה [ההודעה נחתכה]

תכנון לוגיקה:



תכנון קידוד:



0000105		Address of label STR	0000000000000001111010010
0000106	inc r6		00010100000111100001100

נספח - דרישות הפרויקט:

פרויקט הגמר יכלול בהגשתו:

- קובץ זה - אפיון ותכנון הפרויקט.
- קבצי המקור המכילים סיומות c. או h.
- קובץ הרצה עבור מערכת אובונטו
- קובץ makefile עם הדגלים המתאימים, התכנית תתקמפל ללא הערות/אזהרות.
- דוגמאות הרצה
  - **הרצה תקינה**
    - קבצי קלט בשפת אסמבלי שידגימו שימוש במגוון פעולות וטיפוסי הנתונים של השפה
    - קבצי פלט שנוצרו מהפעלת האסמבלר על קבצי הקלט
  - **הדגמת שגיאות**
    - קבצי קלט המדגימים מגוון רחב של שגיאות אסמבלי – כך שלא יווצרו קבצי פלט
    - תדפיסי מסך המראים את הודעות השגיאה שהאסמבלר פולט

דגשים בכתיבת הקוד:

- הגשת הפרויקט מתבצעת תוך חילוק של הקוד לקבצים מתאימים לפי משימות
- הפרדה בין הגישה למבנה הנתונים לבין המימוש שלו.
  - לדוגמא: פונקציית טיפול בטבלה- הטבלה יכולה להיות ממומשת ע"י מערך/רשימה מקושרת
- קריאות וקונבנציות – שמות משמעותיים, הזחות עקביות, הפרדה בין קטעי קוד...
- מקוריות – לא להיעזר בספריות חיצוניות או במקורות חיצוניים.

רקע כללי לפרויקט:

- יחידת העיבוד המרכזית (המעבד - CPU) יכולה לבצע מגוון פעולות פשוטות, הנקראות הוראות מכונה, הוראות המכונה ושילובים שלהן הן המרכיבות תוכנית כפי שהיא טעונה לזיכרון בזמן ריצתה.
- המעבד יודע לבצע קוד שנמצא בפורמט של שפת מכונה. זהו רצף של ביטים, המהווים קידוד בינארי של סדרת הוראות המכונה המרכיבות את התוכנית.
- שפת אסמבלי היא שפת תכנות המאפשרת לייצג את הוראות המכונה בצורה סימבולית קלה ונוחה יותר לשימוש.
- כמובן שיש צורך לתרגם את הייצוג הסימבולי לקוד בשפת מכונה, כדי שהתוכנית תוכל לרוץ במחשב. התרגום מתבצע באמצעות כלי הנקרא **assembler**.
- תפקידו של האסמבלר הוא לבנות קובץ המכיל קוד מכונה, מקובץ נתון של תכנית שכתובה באסמבלי (בשלבים של קישור וטעינה לא נעסוק בפרויקט)
- **המשימה בפרויקט היא: לכתוב אסמבלר – תכנית המתרגמת לשפת מכונה עבור שפת אסמבלי שיוגדר כאן בפרויקט.**

הגדרת מבנה המחשב הדמיוני:

- מעבד – cpu המכיל רגיסטרים
  - למעבד 8 רגיסטרים כלליים בשמות r0-r7 (שמות הרגיסטרים נכתבים תמיד עם אות "r" קטנה)
  - גודל כל רגיסטר 24 סיביות
    - הסיבית הכי פחות משמעותית – סיבית 0
    - הסיבית הכי משמעותית – סיבית 23
  - קיים רגיסטר בשם PSW (program status word) המכיל מספר דגלים המאפיינים את מצב הפעילות במעבד בכל רגע נתון (בתיאור הוראות המכונה יש הסברים לגבי השימוש בדגלים אלו)
- זיכרון RAM
  - חלק מהזיכרון משמש גם כמחסנית
  - גודל הזיכרון הוא 2 בחזקת 21 תאים.
  - לכן הכתובות הן 0 ועד 2 בחזקת 21 פחות 1
  - לתא בזיכרון נקרא "מילה", הסיביות בכל מילה ממוספרות כמו ברגיסטר.
- מחשב זה עובד עם:
  - רק עם מספרים שלמים וחיוביים – אין תמיכה במספרים ממשיים

- הארית‍מטיקה נעשית בשיטת המשלים ל-2
- יש תמיכה בתווים, שמיוצגים בקוד אסקי.

מבנה הוראת מכונה:

- כל הוראת מכונה במודל מורכבת מפעולה ואופרנדים
- מספר האופרנדים הוא בין 0 ל-2 בהתאם לסוג פעולה
- קיים אופרנד מקור ואופרנד יעד
- כל הוראת מכונה מקודדת למס' מילות זיכרון רצופות, החל ממילה אחת ועד למקסימום 3 מילים
  - בהתאם לשיטת המיעון בה נתון כל אופרנד
- בקובץ הפלט שמכיל את קוד המכונה שבונה האסמבלר, כל מילה תקודד בבסיס הקסה.
- במודל המכונה יש 16 פעולות (למרות שניתן לקודד יותר פעולות). שם הפעולה נכתב תמיד באותיות קטנות.
- כל פעולה מיוצגת כך:
  - בקוד המכונה ע"י קוד פעולה (opcode) ופונקציה (func) → באסמבלי באופן סימבולי ע"י שם פעולה
- בכל סוגי הוראות המכונה המבנה של המילה הראשונה הוא:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
opcode						מיעון מקור		רגיסטר מקור			מיעון יעד		רגיסטר יעד			funct					A	R	E

מפרט השדות:

- Opcode – ישנן מספר פעולות עם קוד פעולה זהה, ומה שמבדיל בניהן הוא השדה funct
- Funct – מכיל ערך ייחודי לכל פעולה מקבוצות הפעולות שיש להן אותו קוד פעולה. אם הקוד פעולה משמש לפעולה אחת ה-funct מאופס
- מיעון מקור – מכילות את מספרה של שיטת המיעון של אופרנד המקור
- רגיסטר מקור – מספרו של אגיסטר המקור. אם אין – מאופס
- מיעון יעד – מכילות את מספרה של שיטת המיעון של אופרנד היעד. אם אין – מאופס
- רגיסטר יעד – מספרו של אגיסטר היעד. אם אין – מאופס
- הדגלים A, R, E – ערך הסיבית A היא תמיד 1, והשאר 0.

שיטות מיעון:

קיימות 4 שיטות מיעון –מספרים 0 עד 3

- השימוש בחלקן מצריך מילות מידע נוספות בקוד המכונה חוץ מהמילה הראשונה
- לכל אופרנד של ההוראה נדרשת לכל היותר מילת מידע אחת נוספת. קודם תופיע מילת המידע של אופרנד המקור, ואחריה של אופרנד היעד.
  - בסיביות 0-2 של כל מילת מידע נוספת של ההוראה מקודדת באחד משלושה סוגים של קידוד: הסיביות הן ARE שמציינות מה הסוג –
    - סיבית A (סיבית 2) – מציינת שקידוד המילה הוא מוחלט – Absolute – ואינו מצריך שינוי בשלבי קישור וטעהמסומנות בינה
    - סיבית R (סיבית 1) – מציינת שהקידוד הוא של כתובת פנימית שניתנת להזזה ומצריך שינוי בשלבי הקישור והטעינה.
    - סיבית E (סיבית 0) – מציינת שהקידוד הוא של כתובת חיצונית ומצריך שינוי בשלב הקישור והטעינה.

מספר	שיטת המיעון	תוכן מילת-המידע הנוספת	אופן כתיבת האופרנד	דוגמה
0	מיעון מיידי	מילת-מידע נוספת של ההוראה מכילה את האופרנד עצמו, שהוא מספר שלם בשיטת המשלים ל-2, ברוחב של 21 סיביות, השוכן בסיביות 3-23 של המילה. הסיביות 0-2 של מילת המידע הן השדה A,R,E. במיעון מיידי, ערך הסיבית A הוא 1, ושתי הסיביות האחרות מאופסות.	האופרנד מתחיל בתו # ולאחריו ובצמוד אליו מופיע מספר שלם בבסיס עשרוני.	mov #-1, r2 בדוגמה זו האופרנד הראשון של הפקודה (אופרנד המקור) נתון בשיטת מיעון מיידי. ההוראה כותבת את הערך 1- אל רגיסטר r2.

מספר	שיטת המיעון	תוכן מילת-המידע הנוספת	אופן כתיבת האופרנד	דוגמה
1	מיעון ישיר	מילת-מידע נוספת של ההוראה מכילה כתובת בזיכרון. המילה בכתובת זו בזיכרון היא האופרנד. הכתובת מיוצגת כמספר <u>ללא סימן</u> ברוחב של 21 סיביות, בסיביות 3-23 של מילת המידע. הסיביות 0-2 במילת המידע הן השדה A,R,E. במיעון ישיר, ערך הסיביות האלה תלוי בסוג הכתובת הרשומה בסיביות 3-23. אם זוהי כתובת שמייצגת שורה בקובץ המקור הנוכחי (כתובת פנימית), ערך הסיבית R הוא 1, ושתי הסיביות האחרות מאופסות. ואילו אם זוהי כתובת שמייצגת שורה בקובץ מקור אחר של התוכנית (כתובת חיצונית), ערך הסיבית E הוא 1, ושתי הסיביות האחרות מאופסות.	האופרנד הוא <u>תווית</u> שכבר הוגדרה, או שתוגדר בהמשך הקובץ. ההגדרה נעשית על ידי כתיבת התווית בתחילת השורה של הנחית 'data.' או 'string.', או בתחילת השורה של הוראה, או באמצעות אופרנד של הנחית 'extern.'. התווית מייצגת באופן סימבולי כתובת בזיכרון.	השורה הבאה מגדירה את התווית x: x: .data 23 ההוראה: dec x מקטינה ב-1 את תוכן המילה שבכתובת x בזיכרון (ה"משתנה" x). <u>דוגמה נוספת</u>: ההוראה jmp next מבצעת קפיצה אל השורה בה מוגדרת התווית next (כלומר ההוראה הבאה שתתבצע נמצאת בכתובת next). הכתובת next תקודד בסיביות 3-23 של מילת המידע הנוספת.

מספר	שיטת המיעון	תוכן מילת-המידע הנוספת	אופן כתיבת האופרנד	דוגמה
2	מיעון יחסי	שיטה זו רלוונטית <u>אך ורק</u> להוראות המבצעות קפיצה (הסתעפות) להוראה אחרת. מדובר בקודי-הפעולה הבאים בלבד: jmp, bne, jsr. <u>לא ניתן</u> להשתמש בשיטה זו בהוראות עם קודי-פעולה אחרים. בשיטה זו, יש בקידוד ההוראה מילת מידע נוספת המכילה את מרחק הקפיצה, במילות זיכרון, מכתובת ההוראה הנוכחית (פקודת הקפיצה) אל כתובת ההוראה המבוקשת (ההוראה הבאה לביצוע). מרחק הקפיצה מיוצג כמספר עם סימן בשיטת המשלים ל-2 ברוחב של 21 סיביות, השוכן בסיביות 3-23 של מילת המידע הנוספת. מרחק זה יהיה שלילי במקרה שהקפיצה היא אל הוראה שבכתובת יותר נמוכה, וחיובי במקרה שהקפיצה היא אל הוראה שבכתובת יותר גבוהה. הסיביות 0-2 של מילת המידע הן השדה A,R,E. במיעון יחסי, ערך הסיבית A הוא 1, ושתי הסיביות האחרות מאופסות.	האופרנד מתחיל בתו & ולאחריו ובצמוד אליו מופיע שם של תווית. התווית מייצגת באופן סימבולי כתובת של <u>הוראה בקובץ המקור הנוכחי של התוכנית</u>. ייתכן שהתווית כבר הוגדרה, או שתוגדר בהמשך הקובץ. ההגדרה נעשית על ידי כתיבת התווית בתחילת שורת הוראה. יודגש כי בשיטת מיעון יחסי <u>לא ניתן</u> להשתמש בתווית (כתובת) שמוגדרת בקובץ מקור אחר (כתובת חיצונית).	jmp &next בדוגמה זו, ההוראה מבצעת קפיצה אל השורה בה מוגדרת התווית next (כלומר ההוראה הבאה שתתבצע נמצאת בכתובת next). נניח כי ההוראה jmp שבדוגמה נמצאת בכתובת 500 (עשרוני). כמו כן, נניח כי התווית next מוגדרת בקובץ המקור הנוכחי בכתובת 300 (עשרוני). מרחק הקפיצה אל ההוראה בכתובת next הוא 200-, ומרחק זה יקודד בסיביות 3-23 של מילת המידע הנוספת.

מספר	שיטת המיעון	תוכן מילת-המידע הנוספת	אופן כתיבת האופרנד	דוגמה
3	מיעון רגיסטר ישיר	האופרנד הוא רגיסטר. לשיטת מיעון זו אין מילת מידע נוספת. מספרו של הרגיסטר מקודד במילה הראשונה של ההוראה, בשדה המתאים: רגיסטר מקור/יעד.	האופרנד הוא שם של רגיסטר.	clr r1 בדוגמה זו, ההוראה clr מאפסת את תוכן הרגיסטר r1.

מפרט הוראות המכונה:

הוראות הדורשות 2 אופרנדים:

הוראה	opcode	funct	הפעולה המתבצעת	דוגמה	הסבר הדוגמה
mov	0		מבצעת העתקה של תוכן אופרנד המקור (האופרנד הראשון) אל אופרנד היעד (האופרנד השני).	mov A, r1	העתק את תוכן המשתנה A (המילה שבכתובת A בזיכרון) אל רגיסטר r1.
cmp	1		מבצעת השוואה בין שני האופרנדים. ערך אופרנד היעד (השני) מופחת מערך אופרנד המקור (הראשון), ללא שמירת תוצאת החיסור. פעולת החיסור מעדכנת דגל בשם Z (״דגל האפס״) ברגיסטר הסטטוס (PSW).	cmp A, r1	אם תוכן המשתנה A זהה לתוכנו של רגיסטר r1 אזי הדגל Z (״דגל האפס״) ברגיסטר הסטטוס (PSW) יודלק, אחרת הדגל יאופס.
add	2	1	אופרנד היעד (השני) מקבל את תוצאת החיבור של אופרנד המקור (הראשון) והיעד (השני).	add A, r0	רגיסטר r0 מקבל את תוצאת החיבור של תוכן המשתנה A ותוכנו הנוכחי של r0.
sub	2	2	אופרנד היעד (השני) מקבל את תוצאת החיסור של אופרנד המקור (הראשון) מאופרנד היעד (השני).	sub #3, r1	רגיסטר r1 מקבל את תוצאת החיסור של הקבוע 3 מתוכנו הנוכחי של הרגיסטר r1.
lea	4		lea הוא קיצור (ראשי תיבות) של load effective address. פעולה זו מציבה את המען בזיכרון המיוצג על ידי התווית שבאופרנד הראשון (המקור), אל אופרנד היעד (האופרנד השני).	lea HELLO, r1	המען שמייצגת התווית HELLO מוצב לרגיסטר r1.

הוראות הדורשות אופרנד אחד – השדות של אופרנד המקור יהיו מאופסים:

הוראה	opcode	funct	הפעולה המתבצעת	דוגמה	הסבר הדוגמה
clr	5	1	איפוס תוכן האופרנד	clr r2	הרגיסטר r2 מקבל את הערך 0.
not	5	2	היפוך ערכי הסיביות באופרנד (כל סיבית שערכה 0 תהפוך ל-1 ולהיפך: 1 ל-0).	not r2	כל ביט ברגיסטר r2 מתהפך.
inc	5	3	הגדלת תוכן האופרנד באחד.	inc r2	תוכן הרגיסטר r2 מוגדל ב-1.
dec	5	4	הקטנת תוכן האופרנד באחד.	dec Count	תוכן המשתנה Count מוקטן ב- 1.
jmp	9	1	קפיצה (הסתעפות) בלתי מותנית אל ההוראה שנמצאת במען המיוצג על ידי האופרנד. כלומר, כתוצאה מביצוע ההוראה, מצביע התוכנית (PC) מקבל את כתובת יעד הקפיצה.	jmp &Line	PC←PC+distanceTo(Line) מצביע התכנית מקבל את המען שמחושב על ידי חיבור המרחק לתווית Line עם מען ההוראה הנוכחית, ולפיכך ההוראה הבאה שתתבצע תהיה במען Line.
bne	9	2	bne הוא קיצור (ראשי תיבות) של: branch if not equal (to zero). זוהי הוראת הסתעפות מותנית. אם ערכו של הדגל Z ברגיסטר הסטטוס (PSW) הינו 0, אזי מצביע התוכנית (PC) מקבל את כתובת יעד הקפיצה. כזכור, הדגל Z נקבע באמצעות הוראת cmp.	bne Line	אם ערך הדגל Z ברגיסטר הסטטוס (PSW) הוא 0, אזי PC ← address(Line) מצביע התכנית יקבל את כתובת התווית Line, ולפיכך ההוראה הבאה שתתבצע תהיה במען Line.
jsr	9	3	קריאה לשגרה (סברוטינה). כתובת ההוראה שאחרי הוראת jsr הנוכחית (PC+2) נדחפת לתוך המחסנית שבזיכרון המחשב, ומצביע התוכנית (PC) מקבל את כתובת השגרה. <u>הערה</u>: חזרה מהשגרה מתבצעת באמצעות הוראת rts, תוך שימוש בכתובת שבמחסנית.	jsr SUBR	push(PC+2) PC ← address(SUBR) מצביע התכנית יקבל את כתובת התווית SUBR, ולפיכך, ההוראה הבאה שתתבצע תהיה במען SUBR. כתובת החזרה מהשגרה נשמרת במחסנית.
red	12		קריאה של תו מהקלט הסטנדרטי (stdin) אל האופרנד.	red r1	קוד ה-ascii של התו הנקרא מהקלט ייכנס לרגיסטר r1.
prn	13		הדפסת התו הנמצא באופרנד, אל הפלט הסטנדרטי (stdout).	prn r1	יודפס לפלט התו (קוד ascii) הנמצא ברגיסטר r1

ללא אופרנדים – אופרנד המקור והיעד יהיו מאופסים במילה הראשונה:

הוראה	opcode	הפעולה המתבצעת	דוגמה	הסבר הדוגמה
rts	14	מתבצעת חזרה משיגרה. הערך שבראש המחסנית של המחשב מוצא מן המחסנית, ומוכנס למצביע התוכנית (PC). הערה: ערך זה נכנס למחסנית בקריאה לשגרה ע"י הוראת jsr	rts	PC ← pop() ההוראה הבאה שתתבצע תהיה זו שאחרי הוראת jsr שקראה לשגרה.
stop	15	עצירת ריצת התוכנית.	stop	התוכנית עוצרת מיידית.

מבנה שפת אסמבלי -

מאקרואים:

- קטעי קוד שכוללים משפטים, השימוש הוא פשוט באזכור המאקרו.

```
mcro  a_mc
  inc r2
  mov  A,r1
mcroend
```

<u>תמיכה במאקרו בפרויקט:</u>

הנחות:

- אין במערכת הגדרות מאקרו מקוננות
- לכל שורת מאקרו בקוד המקור קיימת סגירה עם שורת mcroend
- הגדרת מאקרו קיימת לפני הקריאה למאקרו

הנחיות:

- בדיקה של שם המאקרו – אין שם של הוראה / הנחיה
- לבדוק שבשורת ההגדרה ובשורת הסיום אין תווים נוספים
- הקדם אסמבלר יוצר קובץ הכולל פרישה של המאקרו
  - קובץ מקור מורחב הוא קובץ מקור לאחר פרישת מאקרו
  - קיים גם קובץ מקור ראשוני הוא קובץ הקלט למערכת (כולל הגדרת מאקרואים)
- אם נמצאה שגיאה – יש לעצור ולהודיע על השגיאות ולעבור לקובץ המקור הבא במידה וקיים.
  - שגיאות בגוף המאקרו מגלים בשלבים הבאים

משפטים:

סוג המשפט	הסבר כללי
משפט ריק	זוהי שורה המכילה אך ורק תווים לבנים (whitespace), כלומר רק את התווים ' ' ו- '\t' (רווחים וטאבים). ייתכן ובשורה אין אף תו (למעט התו \n), כלומר השורה ריקה.
משפט הערה	זוהי שורה בה התו הראשון הינו ';' (נקודה פסיק). על האסמבלר להתעלם לחלוטין משורה זו.
משפט הנחיה	זהו משפט המנחה את האסמבלר מה עליו לעשות כשהוא פועל על תכנית המקור. יש מספר סוגים של משפטי הנחיה. משפט הנחיה עשוי לגרום להקצאת זיכרון ואתחול משתנים של התכנית, אך הוא אינו מייצר קידוד של הוראות מכונה המיועדות לביצוע בעת ריצת התכנית.
משפט הוראה	זהו משפט המייצר קידוד של הוראות מכונה לביצוע בעת ריצת התכנית. המשפט מורכב משם של הוראה שעל המעבד לבצע, ותיאור האופרנדים של ההוראה.

- כל משפט מופיע בשורה נפרדת וההפרדה בין משפט למשפט הינה באמצעות התו "\n"
- אורכה של כל שורה בקובץ המקור הוא עד 80 תווים

משפט הנחיה:

- משפט המנחה את האסמבלר מה עליו לעשות כשהוא פועל על תבנית המקור
- עשוי לגרום להקצאת זיכרון ואתחול משתנים של התבנית

בעל מבנה:

  - בתחילת המשפט יכולה להופיע הגדרה של תווית כאשר לתווית תחביר חוקי (אופציונלי)

- לאחר מכן מופיע שם ההנחיה שמתחיל בנקודה "."
- כאשר אחריו יופיעו פרמטרים בהתאם להנחייה שהם תווים באותיות קטנות בלבד.

סוגי משפטי ההנחיה:

- ***.data***
  - הנחייה להקצות מקום בתמונת הנתונים שבו יאוחסנו הערכים של הפרמטרים ולקדם את מונה הנתונים בהתאם למספר הערכים.
  - אם בהנחיית Data מוגדרת תווית אז היא מקבלת את ערך מונה הנתונים לפני קידום ומוכנסת אל טבלת הסמלים. (הגדרת שם של משתנה)
  - הפרמטרים הם מספרים שלמים חוקיים המופרדים על ידי פסיק
  - לדוגמא: data 7, -57, +17, 9.
  - הפסיקים אינם חייבים להיות צמודים למספרים (יכולים להופיע רווחים וטאבים בכמות מסוימת או בכלל לא), אבל הפסיק חייב להיות קיים בין המספרים.
  - אסור שיופיע יותר מפסיק אחד בין שני מספרים וגם לא פסיק אחרי האחרון או לפני הראשון.

    כלומר אם נכתוב:

    XYZ: .data 7, –57, +17, 9

    אזי יוקצו בתמונת הנתונים ארבע מילים רצופות שיכילו את המספרים שמופיעים בהנחיה.
    התווית XYZ מזוהה עם כתובת המילה הראשונה.

    אם נכתוב בתכנית את ההוראה:

    mov XYZ, r1

    אזי בזמן ריצת התכנית יוכנס לרגיסטר r1 הערך 7.

    ואילו ההוראה:

    lea XYZ, r1

    תכניס לרגיסטר r1 את ערך התווית XYZ (כלומר הכתובת בזיכרון בה מאוחסן הערך 7).
- ***.string***
  - להנחיה זו יש פרמטר אחד שהוא מחרוזת חוקית כאשר תווי המחרוזת מקודדים לפי ערכי האסקי המתאימים, ומוכנסים אל תמונת הנתונים לפי סדרם, כל תו במילה נפרדת.
  - בסוף המחרוזת יתווסף התו 0/ (הערך המספרי 0) המסמן את סוף המחרוזת.
  - מונה הנתונים של האסמבלר יקודם בהתאם לאורך המחרוזת (בתוספת מקום אחד עבור התו המסיים)
  - אם בשורת ההנחיה מוגדרת תווית אז תווית זו מקבלת את ערך מונה הנתונים (לפני קידום) ומוכנסת לטבלת הסמלית בדומה לdata.

    STR: .string "abcdef"

    מקצה בתמונת הנתונים רצף של 7 מילים, ומאתחלת את המילים לקודי ה-ascii של התווים לפי הסדר במחרוזת, ולאחריהם הערך 0 לסימון סוף מחרוזת. התווית STR מזוהה עם כתובת התחלת המחרוזת.
- ***.entry***
  - הפרמטר הוא שם של תווית שמוגדרת בקובץ המקור הנוכחי
  - מטרת ההנחייה היא לאפיין את תווית זו באופן שיאפשר לקוד הנמצא בקבצי מקור אחרים להשתמש בה גם כאופרנד של הוראה.

    לדוגמה, השורות:

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

    מודיעות לאסמבלר שאפשר להתייחס בקובץ אחר לתווית HELLO המוגדרת בקובץ הנוכחי.

    לתשומת לב: תווית המוגדרת בתחילת שורת entry. הינה חסרת משמעות והאסמבלר **מתעלם** מתווית זו (אפשר שהאסמבלר יוציא הודעת אזהרה).
- ***.extern***
  - הפרמטר הוא שם של תווית שאינה מוגדרת בקובץ מקור הנוכחי
  - המטרה היא להודיע לאסמבלר שהתווית המוגדרת בקובץ מקור אחר, ושניתן להשתמש בה כאופרנד של הוראה

לדוגמה, משפט ההנחיה 'extern.' התואם למשפט ההנחיה 'entry' מהדוגמה הקודמת יהיה:

.extern HELLO

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

לתשומת לב: תווית המוגדרת בתחילת שורת extern. הינה חסרת משמעות והאסמבלר **מתעלם** מתווית זו (אפשר שהאסמבלר יוציא הודעת אזהרה).

משפט הוראה:

- מורכב מ3 חלקים: תווית, שם הפעולה, אופרנדים
- אם מוגדרת תווית, אז היא תוכנס לטבלת הסמלים. הערך יהיה מען המילה הראשונה של ההוראה בתוך תמונת הקוד שבונה האסמבלר.
- שם הפעולה תמיד באותיות קטנות והוא אחת מ16 הפעולות שפורטו.
- לאחר שם הפעולה, יופיעו האופרנדים בהתאם לסוג הפעולה. יש להפריד בין שם הפעולה לבין האופרנד הראשון באמצעות רווחים או טאבים.
- כאשר יש יותר מאופרנד אחד, הם מופרדים על ידי פסיק ולא חייבת להיות הצמדה של האופרנדים לפסיק. כל כמות של רווחים 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

-
- סימון המילים בקוד המכונה באמצעות המאפיין "A R E"
  - בכל מילה בקוד המכונה של הוראה (לא של נתונים) האסמבלר מכניס מידע עבור תהליך הקישור והטעינה

שלוש הסיביות בשדה A,R,E יכילו ערכים בינאריים כפי שהוסבר בתיאור שיטות המיעון.
המשמעות של כל ערך מפורטת להלן.

האות 'A' (קיצור של absolute) באה לציין שתוכן המילה אינו תלוי במקום בזיכרון בו ייטען בפועל קוד המכונה של התכנית בעת ביצועה (למשל מילה המכילה אופרנד מיידי).

האות 'R' (קיצור של relocatable) באה לציין שתוכן המילה תלוי במקום בזיכרון בו ייטען בפועל קוד המכונה של התכנית בעת ביצועה (למשל מילה המכילה כתובת של תווית המוגדרת בקובץ המקור).

האות 'E' (קיצור של external) באה לציין שתוכן המילה תלוי בערכו של סמל חיצוני (external) (למשל מילה המכילה כתובת של תווית חיצונית, כלומר תווית שאינה מוגדרת בקובץ המקור).

השדות במשפטים של שפת האסמבלי:

- *תווית*
  - תווית היא ייצוג סימבולי של כתובת בזיכרון
  - סמל שמוגדר בתחילת משפט הוראה או בתחילת הנחייה
  - תווית חוקית מתחילה באות אלפביתית גדולה או קטנה
  - אחריה יש סדרה של אותיות אלפביתיות / ספרות
  - אורך מקסימלי הוא 31 תווים
  - הגדרה של תווית מסתיימת ב ":" (כמובן תו זה אינו חלק מהתווית)
    - התו חייב להיות צמוד לתווית ללא רווחים
  - אסור שאותה תווית תוגדר יותר מפעם אחת
  - אסור שאותו סמל ישמש הן כתווית והן כשם של מאקרו
  - אסור שמילים שמורות של השפה יהיו כתווית

לדוגמה, התוויות המוגדרות להלן הן תוויות חוקיות.

hEllo:
x:
He78902:

- תווית המוגדרת בהנחיות data/string תקבל את ערך מונה הנתונים הנוכחי
- תווית המוגדרת בשורת הוראה תקבל את ערך מונה ההוראות
- מותר במשפט הוראה להשתמש באופרנד שהוא סמל שאינו מוגדר כתווית בקובץ הנוכחי כל עוד הוא מאופיין כחיצוני באמצעות הנחיית extern. כלשהי בקובץ הנוכחי

- *מספר*
  - התמיכה רק בבסיס עשרוני
  - מספר חוקי מתחיל ב "-" או "+" כאשר אחריו סדרה כלשהי של ספרות
- *מחרוזת*
  - מחרוזת חוקית היא סדרת תווי אסקי שמוקפים במרכאות

שלבי טיפול האסמבלר – באופן כללי:

- ***קדם אסמבלר***
  - פרישת מקרואים
    - הפעלת אלגוריתם פרישת מאקרו
    - אם תהליך זה מסתיים בהצלחה יש לעבור לשלב הבא
    - אחרת – יש להציג את השגיאות ולא לייצר קבצים
  - שמירה בקובץ חדש
- ***מעבר על התכנית***
  - שלב ראשון
    - ספירת המקומות בזיכרון אותם תופסות ההוראות
      - אם כל הוראה תיטען בזיכרון למקום העוקב להוראה הקודמת תציין ספירה כזאת את כתובת ההוראה הבאה
      - הספירה נעשית ע"י האסמבלר ומוחזקת במונה IC
        - ערך התחלתי של IC הוא 100
        - מתעדכן בכל שורת הוראה המקצה מקום בזיכרון
        - ה-IC מתעדכן בכל שורת הוראה המקצה מקום בזיכרון, לאחר שהאסמבלר קובע מהו אורך ההוראה, ה-IC מוגדל במספר התאים הנתפסים ע"י ההוראה וכך הוא מצביע על התא הבא שפנוי

    - זיהוי הסמלים (התוויות)
    - מתן לכל סמל ערך מספרי שהוא הכתובת בזיכרון שהסמל מייצג
    - קידוד מתאים של המילה הראשונה עבור משפט הוראה
    - קידוד של מילת מידע נוספת עבור אופרנד מיידי/רגיסטר
    - קידוד מתאים של הנתונים שמתקבלים ממשפטי הנחיות – data,string...
    -
  - שלב שני
    - החלפת שמות הפעולות -> קוד הבינארי השקול להם במודל המחשב
    - החלפת הסמלים בכתובות המתאימות של הזיכרון שם נמצאים הנתונים/ ההוראות לפי הטבלת סמלים
  - הפרדה בין קטע הנתונים לקטע ההוראות
    - בקובץ הפלט (המכונה) תהיה הפרדה של הוראות ונתונים לשני קטעים נפרדים
    - בקובץ הקלט אין חובה שתהיה הפרדה כזו

גילוי שגיאות

אם יש שגיאות – אין קובץ פלט.

הדפסה אל הפלט הסטנדרטי בפורמט " מס שורה: -מניין השורות מתחיל ב1- שגיאה: "

- קדם אסמבלר – פרישת מאקרואים

- בדיקה של שם המאקרו – אין שם של הוראה / הנחיה
- לבדוק שבשורת ההגדרה ובשורת הסיום אין תווים נוספים
  - אם נמצאה שגיאה – יש לעצור ולהודיע על השגיאות ולעבור לקובץ המקור הבא במידה וקיים.
- שגיאות בגוף המאקרו מגלים בשלבים הבאים – אם יש שגיאה היא תוצג כמה פעמים כי זה לאחר פרישת מאקרו

- פעולה שאינה קיימת
- מספר אופרנדים שגוי
- סוג אופרנד שאינו מתאים לפעולה (שיטת מיעון לא חוקית למשל)

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

- שם רגיסטר לא קיים
- וידוא שכל סמל מוגדר פעם אחת בדיוק
- "יותר מדיי אופרנדים" בהוראה שאמור להיות בה אופרנד יחיד

הוצאת קבצי פלט

- בהפעלה של האסמבלר, יש להעביר אליו באמצעות ארגומנטים של שורת הפקודה רשימה של שמות קבצי מקור (אחד או יותר).
- אלו הם קבצי טקסט, ובהם תוכניות בתחביר של שפת האסמבלי שהוגדרה בפרויקט.
- האסמבלר פועל על כל קובץ מקור בנפרד, ויוצר עבורו קבצי פלט כדלקמן:
  - קובץ am המכיל את קוד המקור לאחר שלב קדם האסמבלר (פרישת מאקרואים)
  - קובץ object המכיל את קוד המכונה
  - קובץ externals המכיל פרטים על כל הכתובות בקוד המכונה בהם יש מילת מידע שמקודדת ערך של סמל שהוצהר כחיצוני
  - קובץ entries המכיל פרטים על כל סמל שמוצהר כפנימי (מאופיין בטבלת הסמלים כ-entry)
- שמות קבצי הפלט צריכים להיות מבוססים על שם קובץ הקלט
- אם אין בקובץ המקור אף הנחיית externals, האסמבלר לא יוצר את קובץ הפלט מסוג externals.
- כנל לגבי entries.
- שמות קבצי המקור חייבים להיות עם הסיומת .as
  - צריכה להיות תמיכה ללא ציון הסיומת

קבצי הפלט – הרחבה

- קובץ object
  - האסמבלר בונה את תמונת הזיכרון כך שקידוד ההוראה הראשונה יכנס למען 100 בזיכרון. מיד לאחר קידוד ההוראה האחרונה, מכניסים לתמונת הזיכרון את קידוד הנתונים שנבנו על ידי ההנחיות .string ו-.data
  - הנתונים יוכנסו בסדר באופן הם מופיעים בקובץ המקור.
  - קובץ זה מורכב משורות של טקסט: השורה הראשונה היא כותרת המכילה שני מספרים: האורך הכולל של קטע ההוראות (במילות זיכרון) ואחריו האורך הכול של קטע הנתונים (במילות זיכרון) כאשר בין שני המספרים יש רווח אחד.
  - שאר השורות מכילות את תוכן הזיכרון – בכל שורה שני מספרים: כתובת של מילה בזיכרון, ותוכן המילה. כל המספרים הם בבסיס 2 הייחודי שהוגדר.
- קובץ entries
  - קובץ זה בנוי משורות טקסט. כל שורה מכילה שם של סמל שהוגדר כentry ואת ערכו, כפי שנמצא בטבלת הסמלים. הערכים מיוצגים בבסיס 2 הייחודי שהוגדר.
- קובץ externals
  - קובץ זה בנוי משורות טקסט. כל שורה מכילה שם של סמל שהוגדר כexternal וכתובת בקוד המכונה בה יש קידוד של אופרנד המתייחס לסמל זה.

- מאחר וייתכן כי יש מספר כתובות בקוד בהם יש התייחסות לסמל זה – לכל התייחסות תהיה שורה נפרדת (הצגה בבסיס 2 הייחודי)
- קובץ am
  - הקובץ של קוד המקור לאחר תהליך קדם האסמבלי

אלגוריתם לפרישת מאקרו:

תכנית שיודעת לפתוח תיקייה וקוראת מקובץ fget וכו

1) קרא את השורה הבאה מקובץ המקור. אם נגמר הקובץ, עבור ל- 9 (סיום).
2) האם השדה הראשון הוא שם מאקרו המופיע בטבלת המאקרו (כגון a_mc)? אם כן, החלף את שם המאקרו והעתק במקומו את כל השורות המתאימות מהטבלה לקובץ, חזור ל 1. אחרת, המשך.
3) האם השדה הראשון הוא "mero" (התחלת הגדרת מאקרו)? אם לא, עבור ל- 6.
4) הדלק דגל "יש macro"
5) הכנס לטבלת שורות מאקרו את שם המאקרו (לדוגמה a_mc).
6) קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עבור ל- 9 (סיום). אם דגל "יש mero" דולק ולא זוהתה תווית mcroend הכנס את השורה לטבלת המאקרו ומחק את השורה הנייל מהקובץ. אחרת (לא מאקרו) חזור ל 1.
7) האם זוהתה תווית meroend? אם כן, מחק את התווית מהקובץ והמשך. אם לא, חזור ל- 6.
8) כבה דגל "יש mcro ". חזור ל- 1. (סיום שמירת הגדרת מאקרו).
9) סיום: שמירת קובץ מאקרו פרוש.

שלד של האסמבלר – לדוגמא מההנחיות:

- האסמבלר מחזיק מערך הוראות ומערך נתונים שנותנים תמונה של זיכרון המכונה – כלומר בגודל 24 סיביות
- מוני ההוראות והנתונים מצביעים על המקום הבא הפנוי במערכים בהתאמה. כאשר מתחיל האסמבלר לעבור על קובץ המקור שני מונים אלו מקבלים ערך התחלתי
  - IC – מונה ההוראות
  - DC – מונה הנתונים
    - קוד המכונה יתאים לטעינה לזיכרון החל מכתובת 100
- טבלה symbol-table שבה נאספות כל התוויות בהן נתקל האסמבלר במהלך המעבר על הקובץ.
  - לכל סמל נשמרים שמו, ערכו, ומאפיינים – code,data,extern,entry

1. קדם אסמבלר
   1.1. פרישת מקרואים
      1.1.1. הפעלת אלגוריתם פרישת מאקרו
      1.1.2. אם תהליך זה מסתיים בהצלחה יש לעבור לשלב הבא
      1.1.3. אחרת – יש להציג את השגיאות ולא לייצר קבצים
   1.2. שמירה בקובץ חדש
2. שלב ראשון
   2.1. אתחול ic=100 dc=0
   2.2. קרא את השורה הבאה מקובץ המקור, אם נגמר קובץ המקור עבור ל-19
   2.3. אם השורה היא ריקה/הערה חזור ל-2
   2.4. האם השדה הראשון בשורה הוא סמל? אם לא עבור ל-5
   2.5. הדלק דגל "יש סמל"
   2.6. האם זוהי הנחיה לאחסון נתונים? אם לא עבור ל-8
   2.7. אם יש הגדרת סמל הכנס לטבלת סמלים עם המאפיין data. ערכו יהיה dc (אם הסמל כבר בטבלה-שגיאה)
   2.8. זהה את סוג הנתונים קודד בזיכרון ועדכן את מונה הנתונים לאורכם. אם זה data. אז האסמבלר קורא את רשימת המספרים ומכניס כל מספר אל מערך הנתונים.
   אם זה string. אז קודי האסקי של התווים הם אלו המוכנסים למערך הנתונים כאשר כל תו בתא אחר. לבסוף מוכנס התו אפס וגם הוא תופס מקום (כלומר מונה הנתונים קודם באורך המחרוזת + 1). חזור ל-2.
   2.9. האם זו הנחיית entry או extern? אם לא עבור ל-11. אם זוהי הנחיית entry חזור ל-2
   2.10. אם זו הנחיית extern הכנס את הסמל המופיע כאופרנד של ההנחייה לתוך טבלת הסמלים עם הערך 0, ועם המאפיין של external. חזור ל-2

2.11. התקבלה שורת הוראה – אם יש הגדרת סמל, הכנס לטבלת הסמלים עם המאפיין code. ערכו של הסמל יהיה ic (אם הסמל כבר בטבלה – שגיאה)
2.12. חפש את שם הפעולה בטבלת שמות הפעולות (אם לא נמצא-שגיאה)
2.13. נתח את מבנה האופרנדים וחשב מהו מס המילים הכולל שתופסת ההוראה בקוד המכונה – נקרא לו L
2.14. בנה את הקוד הבינארי של המילה הראשונה של ההוראה ושל כל מילת מידע נוספת המקודדת אופרנד במיעון מיידי
2.15. שמור את הערכים ic ו-L יחד עם נתוני קוד המכונה של ההוראה
2.16. עדכן ic=ic+L וחזור ל-2
2.17. שמור מקום במערך ההוראות עבור מילים נוספות הנדרשות שעדיין לא קודדו
2.18. אם בשורת ההוראה קיימת תווית, אז התווית מוכנסת אל טבלת הסמלים תחת השם המתאים כאשר ערך התווית הוא מונה ההוראות לפני קידוד ההוראה.
2.19. הסתיימה קריאת קובץ המקור. אם נמצאו שגיאות במעבר הראשון לעצור כאן
2.20. שמירת הערכים הסופיים של IC, DC (נקרא להם ICF,DCF) נשתמש בהם לבניית קבצי הפלט
2.21. עדכן בטבלת הסמלים את ערכו של כל סמל המאופיין כ-data, ע"י הוספת הערך ICF+100 (יש הסבר בסוף עמ' 27 בהוראות)

3. שלב שני
3.1. קרא את השורה הבאה מקובץ המקור. אם נגמר עבור ל-7
3.2. אם השדה הראשון בשורה הוא סמל דלג עליו
3.3. האם זוהי הנחיית data. או string. או extern.? אם כן חזור ל-1
3.4. האם זוהי הנחיית entry? אם לא-עבור ל-6
3.5. הוסף בטבלת הסמלים את המאפיין entry למאפייני הסמל המופיע כאופרנד של ההנחייה (אם לא נמצא בסמלים-שגיאה)
3.6. השלמת קידוד בינארי של מילות המידע של האופרנדים בהתאם לשיטות מיעון
3.6.1. לכל אופרנד המכיל סמל מצא את ערכו בטבלת סמלים (אם לא נמצא-שגיאה).
3.6.2. אם הסמל הוא external הוסף את כתובת המידע הרלוונטית לרשימת מילות מידע שמתייחסות לסמל חיצוני לפי הצורך לחישוב הקידוד והכתובות אפשר להיעזר בערכים ic, L של ההוראה כפי שנשמרו במעבר הראשון
3.7. קובץ המקור נקרא בשלמותו. אם נמצאו שגיאות – לעצור כאן
3.8. בניית קבצי הפלט