

האוניברסיטה הפתוחה

20465

מעבדה בתכנות מערכות

חוברת הקורס – סתיו 2019א

כתבה: מיכל אבימור

אוקטובר 2018 – סמסטר סתיו – תשע"ט

פנימי – לא להפצה.

© כל הזכויות שמורות לאוניברסיטה הפתוחה.

תוכן העניינים

א	אל הסטודנט
ג	1. לוח זמנים ופעילויות
ה	2. תיאור המטלות
ז	3. התנאים לקבלת נקודות זכות
1	ממ"ן 11
5	ממ"ן 12
7	ממ"ן 22
15	ממ"ן 23
17	ממ"ן 14

אל הסטודנט,

אני מקדמת את פניך בברכה, עם הצטרפותך אל הלומדים בקורס "מעבדה בתכנות מערכות". בחוברת זו תמצא את הדרישות לקבלת נקודות זכות בקורס, לוח הזמנים ומטלות הקורס.

לקורס קיים אתר באינטרנט בו תמצאו חומרי למידה נוספים, אותם מפרסם/מת מרכז/ת ההוראה. בנוסף, האתר מהווה עבורכם ערוץ תקשורת עם צוות ההוראה ועם סטודנטים אחרים בקורס. פרטים על למידה מתוקשבת ואתר הקורס, תמצאו באתר שה"ם בכתובת:

<http://telem.openu.ac.il>

מידע על שירותי ספרייה ומקורות מידע שהאוניברסיטה מעמידה לרשותכם, תמצאו באתר

הספריה באינטרנט www.openu.ac.il/Library.

קורס זה הינו קורס מתוקשב. מידע על אופן ההשתתפות בתקשוב ישלח לכל סטודנט באופן אישי. ניתן להפנות שאלות בנושאי חומר הלימוד, והממ"נים לקבוצת הדיון של הקורס. בנוסף יופיעו שם הודעות ועדכונים מצוות הקורס. כניסה תכופה לאתר הקורס ולקבוצת הדיון שלה, מאפשרת לך להתעדכן בכל המידע, ההבהרות וכו' במסגרת הקורס.

ניתן לפנות אלי בשעות הייעוץ שלי (יפורסמו בהמשך באתר) או מחוץ לשעות הקבלה, באמצעות email, לכתובת: michav@openu.ac.il, ואשתדל לענות בהקדם.

לתשומת לב הסטודנטים הלומדים בחו"ל:

למרות הריחוק הפיסי הגדול, נשתדל לשמור אתכם על קשרים הדוקים ולעמוד לרשותכם ככל האפשר.

הפרטים החיוניים על הקורס נכללים בחוברת הקורס וכן באתר הקורס. מומלץ מאוד להשתמש באתר הקורס ובכל אמצעי העזר שבו וכמובן לפנות אלינו במידת הצורך.

אני מאחלת לך לימוד פורה ומהנה.

בברכה,

מיכל אבימור
מרכזת ההוראה בקורס.

1. לוח זמנים ופעילויות (2019א/20465)

שבוע הלימוד	תאריכי שבוע הלימוד	יחידת הלימוד המומלצת	מפגשי ההנחיה*	תאריך אחרון למשלוח הממ"ן (למנחה)
1	19.10.2018-14.10.2018	ספר C פרקים 1-2-3	מפגש ראשון	
2	26.10.2018-21.10.2018	ספר C פרקים 1-2-3		
3	2.11.2018-28.10.2018	ספר C פרק 4	מפגש שני	
4	9.11.2018-4.11.2018	ספר C פרק 4		
5	16.11.2018-11.11.2018	ספר C פרק 5	מפגש שלישי	ממ"ן 11 11.11.2018
6	23.11.2018-18.11.2018	ספר C פרק 5		
7	30.11.2018-25.11.2018	ספר C פרק 6	מפגש רביעי	
8	7.12.2018-2.12.2018 (ב-ו חנוכה)	ספר C פרק 6		ממ"ן 12 2.12.2018

* התאריכים המדויקים של המפגשים הקבוצתיים מופיעים ב"לוח מפגשים ומנחים".

לוח זמנים ופעילויות - המשך

שבוע הלימוד	תאריכי שבוע הלימוד	יחידת הלימוד המומלצת	מפגשי ההנחיה*	תאריך אחרון למשלוח הממ"ן (למנחה)
9	14.12.2018-9.12.2018 (א-ב חנוכה)	ספר C פרק 6		
10	21.12.2018-16.12.2018	ספר C פרק 7	מפגש חמישי	
11	28.12.2018-23.12.2018	ספר C פרק 7		ממ"ן 22 23.12.2018
12	4.1.2019-30.12.2018	ספר C פרק 8 + פרויקט	מפגש שישי	
13	11.1.2019-6.1.2019	פרויקט וחזרה		ממ"ן 23 6.1.2019
14	18.1.2019-13.1.2019	פרויקט וחזרה	מפגש שביעי	ממ"ן 14** 10.3.2019

מועדי בחינות הגמר יפורסמו בנפרד

*התאריכים המדויקים של המפגשים הקבוצתיים מופיעים ב"לוח מפגשים ומנחים".

**** לא תינתן דחייה בהגשת הפרויקט, פרט למקרים חריגים של מילואים או מחלה ממושכת, במקרים אלו יש לתאם את מועד ההגשה עם צוות הקורס.**

2. תיאור המטלות

על מנת לתרגל את החומר הנלמד ולבדוק את ידיעותיך, עליך לפתור את המטלות המצויות בחוברת המטלות.

רוב המטלות בקורס זה הן **מטלות חובה**, והן בעיקרן תוכניות מחשב. שתי מטלות הן רשות. להלן מספרי המטלות ומשקליהן:

ממ"ן	משקל	פרקים
11	4 (ממ"ן חובה)	3,2,1
12	5 (ממ"ן חובה)	5,4
22	8 (ממ"ן רשות)	6,5,4
23	12 (ממ"ן רשות)	8,7,6
14	31 (ממ"ן חובה)	פרויקט גמר

עליך להגיש במהלך הקורס את כל מטלות החובה. את התשובות לממ"נים יש להגיש באמצעות מערכת המטלות (במקרים מיוחדים ניתן להגיש את המטלות באמצעות הדואר או הגשה ישירה למנחה במפגשי ההנחיה. במקרה כזה יש לתאם את הדבר עם הבודק).

יש להגיש את קבצי המקור (.h, .c), קבצי ההרצה, קבצי הסביבה המתאימים (כולל קבצי MAKEFILE), קבצי קלט וקבצי פלט (או צילומי מסך, אם לא נדרשו הקבצים הנ"ל).

הנחיות לכתיבת מטלות וניקודן

ניקוד המטלות יעשה לפי המשקלים הבאים:

א. ריצה - 20%
התכנית עובדת על פי הדרישות בתרגיל, תוך השגת כל המטרות שהוגדרו. התכנית עוברת קומפילציה ללא הערות.

ב. תיעוד - 20%

התיעוד ייכתב בתוך הקוד. אין להוסיף הערות בקבצים נפרדים.

התיעוד יכלול:

- הערה בראש תכנית שתכלול תיאור תמציתי של מטרות התכנית, כיצד מושגת מטרה זו, תיאור המודלים והאלגוריתם, קלט/פלט וכל הנחה שהנכם מניחים.
- לכל מציג (אב-טיפוס) prototype של פונקציה (בקובץ ה-header הצמוד לקוד), יוצמד תיאור של קלט/פלט, ופעולת הפונקציה. **מטרה:** זהו קובץ היצוא ועל כן עליו להסביר למי שאין לו גישה לקוד איך עליו להשתמש בפונקציה.
- לפני הכותרת (header) של כל פונקציה יבוא תיאור של פעולתה, הנחות ואלגוריתם.

מטרה : התיעוד לפני כל פונקציה נועד לתת היכרות ראשונית, לפעולת הפונקציה, תוך פירוט כיצד הפונקציה עושה זאת. תיעוד זה אמור לאפשר לקורא את הקוד (שלא כתב את הקוד), להבין את הקוד.

4) לכל משתנה יהיה שם משמעותי ויוצמד אליו תיעוד לגבי תפקודו בתכנית. i,j,k - משמשים בד"כ כשמות אינדקסים ואין צורך לתעד אותם.

5) לא יופיעו "מספרי קסם" בגוף התכנית למעט 0,1 לאיתחול משתני לולאות. יש להשתמש בקבועים בעלי שמות משמעותיים שיכתבו באותיות גדולות, ויתועדו בשלב ההגדרה. כל טיפוס מורכב יוגדר כ- typedef ויתועד. נהוג לקרוא לטיפוסים מורכבים בשמות משמעותיים ולהשתמש באותיות גדולות.

6) יש להשתמש בשמות משמעותיים ל: פונקציות, מקרואים, משתנים, קבועים, הגדרת טיפוסים וקבצים.

7) יש להקפיד על קריאות ובהירות תוך שימוש באינדנטציה (היסח) מסודרת ואחידה.

ג. תכנות - 40%

יש להקפיד על כתיבה מסודרת ומודולרית של קוד :

- חלוקה לקבצים - כשלכל קובץ מוצמד קובץ header אם צריך (כאשר נדרש בתרגיל).

- חלוקה לפונקציות.

- שימוש במקרואים.

- שימוש נכון ב-MAKEFILE, (במיוחד כאשר אתם נדרשים לחלק את התוכנית למספר קבצים, במסגרת הממ"ן).

- הסתרת אינפורמציה - ושימוש בהפשטת מידע.

- הימנעות ככל שניתן משימוש במשתנים גלובליים.

- שימוש מירבי ונכון במלוא הכלים שמעמידה השפה לרשותכם.

- קוד אלגנטי ולא מסורבל.

ד. יעילות התכנית והתרשמות כללית - 20%

המשקלים הנ"ל מהווים קו מנחה לחלוקת הנקודות. מובן שתהיה התייחסות לכל תכנית לגופה, בהתאם למידת המורכבות של התרגיל.

ינתנו קנסות במיקרים הבאים :

- אי הגשת קבצי סביבה - MAKEFILE – 20 נקודות.
- עבור אותם ממ"נים בהם מוגדר שם קובץ, פונקציה, או פרמטר, שימוש בשם שונה מזה המוגדר בממ"ן – 10 נקודות.

לתשומת לבך : חל איסור מוחלט של הכנה משותפת של מטלות או העתקת מטלות. תלמיד שייתפס באחד מאיסורים אלה ייענש בהתאם לנאמר בתקנון המשמעת נספח 1 בידיעון של האו"פ. רק את ממ"ן 14 מותר להגשה בזוגות (לא ניתן להגיש בשלושות!), כאשר שני הסטודנטים המגישים שיכים לאותה קבוצת לימוד.

3. התנאים לקבלת נקודות זכות בקורס

- א. להגיש את מטלות החובה בקורס (11, 12) וכן את פרויקט הגמר (14).
- ב. ציון של לפחות 60 נקודות בבחינת הגמר.
- ג. ציון סופי בקורס של 60 נקודות לפחות.

לתשומת לבכם!

כדי לעודדכם להגיש לבדיקה מספר רב של מטלות הנהגנו את ההקלה שלהלן:

אם הגשתם מטלות מעל למשקל המינימלי הנדרש בקורס, **המטלות** בציון הנמוך ביותר, שציוניהן נמוכים מציון הבחינה (**עד שתי מטלות**), לא יילקחו בחשבון בעת שקלול הציון הסופי.

זאת בתנאי שמטלות אלה **אינן חלק מדרישות החובה בקורס** ושהמשקל הצבור של המטלות האחרות שהוגשו, מגיע למינימום הנדרש.

זכרו! ציון סופי מחושב רק לסטודנטים שעברו את בחינת הגמר בציון 60 ומעלה והגישו מטלות כנדרש באותו קורס.

מטלת מנחה (ממ"ן) 11

הקורס: 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטלה: פרקים 1,2,3

משקל המטלה: 4 נקודות (חובה)

מספר השאלות: 2

מועד אחרון להגשה: 11.11.2018

סמסטר: 2019א'

קיימות שתי חלופות להגשת מטלות:

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

יש לקמפל עם דגלים מקסימליים, לקבלת כל האזהרות: -Wall -ansi -pedantic. יש להגיש את קבצי המקור (.c, .h), קבצי ההרצה (את קבצי o. אין צורך לצרף), קבצי הסביבה המתאימים (כולל קבצי makefile), וכן קבצי קלט ותדפיסי מסך או קבצי פלט (לפי הנחיות במטלה/במפגש/באתר). כל תכנית תהיה בתיקיה נפרדת. נדרש ששם התיקיה ושם הקובץ לריצה יהיו כשם הקובץ המכיל את הפונקציה main, ללא הסיומת .c. יש להגיש תכניות מלאות (בין השאר מכילות main), הניתנות להידור והרצה, ומאפשרות בדיקה של כל תוצאות הריצה המגוונות ללא צורך בשינויים כלשהם בקוד התכנית. את המטלה יש להגיש בקובץ zip. לאחר ההגשה, יש להוריד את המטלה משרת האו"פ למחשב האישי, ולבדוק שהקבצים אכן הוגשו באופן תקין.

שאלה 1 (תכנית ראשית בקובץ my_add.c) (50 נקודות)

עליכם לכתוב תכנית המכילה פונקציה:

```
unsigned int my_add(unsigned int a, unsigned int b)
```

המבצעת סימולציה של חיבור בבסיס 2, בין שני האופרנדים a, b.

לדוגמה, אם a הוא 42, ובייצוג בינארי 000000000101010

ואם b הוא 123, ובייצוג בינארי 0000000001111011

אז על הפונקציה לחבר שני מספרים אלה באופן הבא:

תחילה, מחברים את שתי הספרות הבינאריות הימניות ביותר. תוצאת חיבור זה היא $1=0+1$, ואין נשא. עוברים לזוג הספרות הבינאריות השני מימין. כאן תוצאת החיבור היא $10=1+1$ (בבסיס 2), כלומר הספרה 0 והנשא 1. את הנשא מעבירים לטור הבא, ומחברים אותו עם זוג הספרות הבינאריות השלישי מימין, וכך הלאה עד הקצה השמאלי של המספרים.

על הפונקציה להדפיס באופן נאה בבסיס 2 את האופרנדים ואת תוצאת החיבור. כמו כן, הפונקציה תחזיר את תוצאת החיבור כמספר מטיפוס unsigned int.

לתשומת לב: אין לבצע חיבור "רגיל" של המשתנים a, b, אלא לממש חיבור פרטני בבסיס 2 כמתואר לעיל.

רמז: אפשר לגשת לכל סיבית של מספר מטיפוס unsigned int באמצעות פעולות מתמטיות מגוונות של שפת C.

בנוסף, עליכם לכתוב תכנית ראשית (הפונקציה main), שתבצע קלט של שני האופרנדים, ותפעיל את הפונקציה my_add על נתוני הקלט. כמו כן, לאחר החזרה מהפונקציה, התכנית הראשית תדפיס באופן נאה בבסיס 10 את האופרנדים, ואת הסכום שהוחזר מהפונקציה.

נתוני הקלט הם שני מספרים שלמים אי-שליליים בבסיס 10, כל מספר ברוחב של עד 6 ספרות עשרוניות משמעותיות. מותר להניח שהקלט תקין.

הקלט לתכנית הוא מ-stdin, ויכול להגיע מהמקלדת או מקובץ (באמצעות redirection בעת הפעלת התכנית). לנוחיותכם, הכינו מספר קבצי קלט והשתמשו בהם שוב ושוב לדיבוג התכנית.

על התכנית להדפיס הודעת בקשה ידידותית לקלט. כמו כן יש להדפיס באופן נאה את הנתונים כפי שנקלטו, וזאת לפני הקריאה לפונקציה. באופן זה, הקלט יוצג גם כאשר הוא מגיע מקובץ.

חובה לצרף להגשה מספר הרצות בדיקה, המדגימות את פעולת התכנית על מגוון נתוני קלט. יש להגיש תדפיסי מסך (או קבצי פלט) של כל ההרצות. במידה ותשתמשו בקבצי קלט, יש להגיש גם קבצים אלה.

שאלה 2 (תכנית ראשית בקובץ palindrome.c) (50 נקודות)

עליכם לכתוב תכנית המכילה פונקציה:

```
int palindrome(char s[ ]);
```

המקבלת כפרמטר מחרוזת תוים s, ומחזירה 1 אם s פלינדרום, 0 אחרת.

תזכורת: מילה היא פלינדרום אם כאשר קוראים את המילה משמאל לימין או מימין לשמאל, מתקבלת אותה המילה. לדוגמא: המילים אמא, אבא הן פלינדרומים. המילים סבא, סבתא אינן פלינדרומים.

על הפונקציה להתעלם מרווחים וטאבים (tab) במחרוזת s. כמו כן, יש להתעלם מהתו \0 המסיים את המחרוזת. אותיות קטנות וגדולות נחשבות שונות זו מזו.

לדוגמה :

עבור הפרמטר :	a b c b a	נקבל 1,
עבור הפרמטר :	never odd or even	נקבל 1,
וכן עבור הפרמטר :	abba	נקבל 1.
לעומת זאת עבור :	Abba	נקבל 0,
וכן עבור :	a bc de	נקבל 0.

בנוסף, עליכם לכתוב תכנית ראשית (הפונקציה main), שתבצע קלט של מחרוזת מהמשתמש, תפעיל את הפונקציה על מחרוזת הקלט, ותדפיס באופן נאה וידידותי את התוצאה.

המחרוזת מועברת בשורת קלט אחת. מותר להניח שהאורך המקסימלי של שורת הקלט הוא 80 תווים (לא כולל התו \n, שאינו נחשב חלק מהמחרוזת).

הקלט לתכנית הוא מ-stdin, ויכול להגיע מהמקלדת או מקובץ (באמצעות redirection בעת הפעלת התכנית). לנוחיותכם, הכינו מספר קבצי קלט, והשתמשו בהם שוב ושוב כדיבוג התכנית.

על התכנית להדפיס הודעת בקשה ידידותית לקלט. כמו כן יש להדפיס באופן נאה את המחרוזת כפי שנקלטה, וזאת לפני הקריאה לפונקציה. באופן זה, הקלט יוצג גם כאשר הוא מגיע מקובץ.

חובה לצרף להגשה מספר הרצות בדיקה, המדגימות את פעולת התכנית על מגוון נתוני קלט. יש להגיש תדפיסי מסך (או קבצי פלט) של כל ההרצות. במידה ותשתמשו בקבצי קלט, יש להגיש גם קבצים אלה.

להזכירכם : לא תנתן דחייה בהגשת הממ"ן, פרט למקרים מיוחדים כגון מילואים או מחלה, במקרים אלו יש לקבל אישור הגשה מצוות הקורס.

בהצלחה!

מטלת מנחה (ממ"ן) 12

הקורס: 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטלה: פרקים 4,5 ובאופן חלקי 6

מספר השאלות: 1 משקל המטלה: 5 נקודות (חובה)

סמסטר: 2019' מועד אחרון להגשה: 2.12.2018

קיימות שתי חלופות להגשת מטלות:

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

יש לקמפל עם דגלים מקסימליים, לקבלת כל האזהרות: Wall-ansi-pedantic. יש להגיש את קבצי המקור (.c, .h), קבצי ההרצה (את קבצי .o אין צורך לצרף), קבצי הסביבה המתאימים (כולל קבצי makefile), וכן קבצי קלט ותדפיסי מסך או קבצי פלט (לפי הנחיות במטלה/במפגש/באתר). כל תכנית תהיה בתיקה נפרדת. נדרש ששם התיקה ושם הקובץ לריצה יהיו כשם הקובץ המכיל את הפונקציה main, ללא הסיומת .c. יש להגיש תכניות מלאות (בין השאר מכילות main), הניתנות להידור והרצה, ומאפשרות בדיקה של כל תוצאות הריצה המגוונות ללא צורך בשינויים כלשהם בקוד התכנית. את המטלה יש להגיש בקובץ zip. לאחר ההגשה, יש להוריד את המטלה משרת האו"פ למחשב האישי, ולבדוק שהקבצים אכן הוגשו באופן תקין.

שאלה 1 (תכנית ראשית בקובץ my_set.c)

עליכם לכתוב תוכנית המקבלת בקלט רשימת ערכים שלמים (מטיפוס int). מרשימת ערכים אלו עליכם לבנות קבוצה.

תזכורת: על פי הגדרה, קבוצה (set) היא אוסף איברים ברי מניה, שכולם שונים זה מזה.

על התוכנית לקלוט מהמשתמש את רשימת הערכים, לבנות מהם את הקבוצה, ולבסוף להדפיס את איברי הקבוצה לפי סדר הגעתם. ניתן להשתמש בכל מבנה נתונים שתבחרו לשם מימוש הקבוצה. עליכם להסביר באמצעות הערות בתוכנית מדוע בחרתם מבנה נתונים זה. כאמור, כל איבר בקבוצה הוא מספר מטיפוס int.

ניתן להניח תקינות של הקלט, אך ייתכנו ערכים שחוזרים יותר מפעם אחת ברשימת הקלט. עבור ערך כזה (שמופיע יותר מפעם אחת בקלט), הופעתו הראשונה קובעת את מיקומו בהדפסת הפלט.

מותר להניח שמספר האיברים בקבוצה הוא לכל היותר 64.

לעומת זאת, מספר הערכים בקלט אינו מוגבל. הקלט יכול להיות במספר רב של שורות, ובכל שורה יכולים להופיע מספר בלתי מוגבל של ערכים. סוף הקלט יזוהה לפי הערך EOF שמוחזר מפונקציית הספרייה הסטנדרטית באמצעותה מבוצע הקלט (בקלט מהמקלדת, ניתן להכניס EOF על ידי הקלדת ctrl-d בתחילת שורה חדשה).

קלט לדוגמה:

```
13 13 13 45 -9 -9 -9 18
18 18 3 4 55 45 66 13 66
```

עבור קלט זה, הפלט יהיה:

```
13 45 -9 18 3 4 55 66
```

עליכם לכתוב שתי פונקציות:

`get_set` מבצעת את הקלט מהמשתמש ובונה את הקבוצה.
`print_set` מדפיסה את איברי הקבוצה בסדר הנדרש ובאופן נאה.

עליכם לקבוע מהם הפרמטרים והערך המוחזר של כל פונקציה, ולתעד זאת בהערות כותרת של הפונקציה. אין להשתמש במשתנים גלובליים.

כמו כן, יש לכתוב תכנית ראשית (הפונקציה `main`) שתקרא ל-`get_set` ולאחר מכן ל-`print_set`.

הקלט לתכנית הוא מ-`stdin`, ויכול להגיע **מהמקלדת או מקובץ** (באמצעות `redirection` בעת הפעלת התכנית). לנוחיותכם, הכינו מספר קבצי קלט והשתמשו בהם שוב ושוב לדיבוג התכנית.

על התכנית להדפיס הודעת בקשה **ידידותית לקלט**. כמו כן יש להדפיס באופן נאה את כל הנתונים **כפי שנקלטו**. באופן זה, הקלט יוצג גם כאשר הוא מגיע מקובץ.

חובה לצרף להגשה מספר הרצות בדיקה המדגימות את פעולת התכנית על קלטים בגדלים ובערכים מגוונים. יש להגיש תדפיסי מסך (או קבצי פלט) של כל הרצות הדוגמה. במידה ותשתמשו בקבצי קלט, יש להגיש גם קבצים אלה.

להזכירכם: לא תנתן דחייה בהגשת הממ"ן, פרט למקרים מיוחדים כגון מילואים או מחלה, במקרים אלו יש לקבל אישור הגשה מצוות הקורס.

בהצלחה!

מטלת מנחה (ממ"ן) 22

הקורס: 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטלה: פרקים 4,5,6

מספר השאלות: 1 משקל המטלה: 8 נקודות (רשות)

סמסטר: 2019 מועד אחרון להגשה: 23.12.2018

קיימות שתי חלופות להגשת מטלות:

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

יש לקמפל עם דגלים מקסימליים, לקבלת כל האזהרות: Wall-ansi-pedantic. יש להגיש את קבצי המקור (.c, .h), קבצי ההרצה (את קבצי o. אין צורך לצרף), קבצי הסביבה המתאימים (כולל קבצי makefile), וכן קבצי קלט ותדפיסי מסך או קבצי פלט (לפי הנחיות במטלה/במפגש/באתר). כל תכנית תהיה בתיקה נפרדת. נדרש ששם התיקה ושם הקובץ לריצה יהיו כשם הקובץ המכיל את הפונקציה main, ללא הסיומת .c. יש להגיש תכניות מלאות (בין השאר מכילות main), הניתנות להידור והרצה, ומאפשרות בדיקה של כל תוצאות הריצה המגוונות ללא צורך בשינויים כלשהם בקוד התכנית. את המטלה יש להגיש בקובץ zip. לאחר ההגשה יש להוריד את המטלה משרת האו"פ למחשב האישי, ולבדוק שהקבצים אכן הוגשו באופן תקין.

שאלה 1 (בקבצים עיקריים complex.h, complex.c, mycomp.c)

עליכם לכתוב תכנית מחשב אינטראקטיבית הקוראת פקודות מהקלט הסטנדרטי, מפענחת ומבצעת אותן. הפקודות עוסקות בפעולות אריתמטיות על מספרים מרוכבים.

תזכורת מספרים מרוכבים:

מספר מרוכב (complex) הוא מספר בן שני חלקים: חלק ממשי וחלק מדומה, כאשר ביניהם רשום הסימן "+" או הסימן "-".

מבנה המספר הוא: $a + bi$ כאשר a החלק הממשי ו- bi החלק המדומה. החלק המדומה הוא מכפלה של שני גורמים: b הוא מספר ממשי, ואילו i מציין את השורש הריבועי של המספר -1,

$$i = \sqrt{-1}$$

דוגמאות של מספרים מרוכבים :

$$-153+24i \qquad 87.5 - (14.3)i \qquad 24.65 + (15.376)i$$

להלן הגדרות של הפעולות החשבוניות הבסיסיות על מספרים מרוכבים :

חיבור של שני מספרים מרוכבים :

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

חסור של שני מספרים מרוכבים :

$$(a + bi) - (c + di) = (a - c) + (b - d)i$$

כפל של מספר מרוכב במספר ממשי :

$$m * (a + bi) = ma + (mb)i$$

כפל של מספר מרוכב במספר מדומה :

$$mi * (a + bi) = mi * a + mi * bi = -mb + (ma)i$$

כפל של מספר מרוכב במספר מרוכב :

$$(a + bi) * (c + di) = a * c + a * di + bi * c + bi * di = (ac - bd) + (ad + bc)i$$

חישוב הערך המוחלט של מספר מרוכב (התוצאה היא מספר ממשי אי-שלילי) :

$$|a + bi| = \sqrt{a^2 + b^2}$$

משימות התוכנית :

עליכם להגדיר, באמצעות שימוש ב typedef את הטיפוס complex אשר מחזיק מספר מרוכב. על מבנה הנתונים שבחרתם להיות יעיל מבחינת כמות זיכרון הנדרשת, ויעיל מבחינת הגישה אליו.

בנוסף, עליכם להגדיר בתכנית הראשית 6 משתנים : A,B,C,D,E,F מטיפוס complex.

בתחילת הריצה, יש לאתחל את כל ששת המשתנים לאפס (הערך המרוכב 0+0i).

כעת, עליכם לבצע פעולות חשבוניות עם מספרים מרוכבים. כל פעולה תופעל באמצעות פקודה שמועברת בקלט לתכנית, כמפורט להלן. בפעולות אלה, אופרנד שהוא משתנה מרוכב יהיה אחד מששת המשתנים שהוגדרו לעיל.

מפרט הפקודות המשמשות כקלטים לתכנית :

1. הצבת מספר מרוכב במשתנה :

read_comp מספר-ממשי-ראשון, מספר-ממשי-שני, שם-משתנה-מרוכב

הפקודה תגרום להצבת ערך מרוכב במשתנה המרוכב ששמו מופיע בפקודה. המספר הממשי הראשון הוא החלק הממשי של המספר המרוכב, והמספר הממשי השני הוא החלק המדומה של המספר המרוכב (החלק המדומה נתון בפקודה ללא הגורם i).

לדוגמה, הפקודה הבאה :

`read_comp A, 5.1, 6.2`

תבצע את ההצבה :

$$A = 5.1 + (6.2)i$$

הערה : זוהי הפקודה היחידה שמשנה את ערכו של משתנה מרוכב בתכנית.

2. הדפסת ערך של משתנה מרוכב :

print_comp שם-משתנה-מרוכב

ערכו של המשתנה המרוכב ששמו ניתן בפקודה יודפס בצורה נאה בפלט.

לדוגמה, הפקודה הבאה :

`print_comp A`

תגרום להדפסת ערך המשתנה A. בהנחה שהפקודה היא בהמשך לדוגמה בסעיף 1, יודפס :

$$5.1 + (6.2)i$$

3. חיבור מספרים מרוכבים :

add_comp שם-משתנה-מרוכב-ב', שם-משתנה-מרוכב-א'

יתבצע חיבור של שני המספרים המרוכבים אשר במשתנים המופיעים בפקודה :

מספר-מרוכב-ב' + מספר-מרוכב-א'

תוצאת הפעולה תודפס לפלט בפורמט זהה לפורמט ההדפסה של סעיף 2.

4. חיסור מספרים מרוכבים :

sub_comp שם-משתנה-מרוכב-ב', שם-משתנה-מרוכב-א'

יתבצע חיסור של המספר המרוכב במשתנה ב' מן המספר המרוכב במשתנה א' :

מספר-מרוכב-ב' – מספר-מרוכב-א'

תוצאת הפעולה תודפס לפלט בפורמט זהה לפורמט ההדפסה של סעיף 2.

5. כפל מספר מרוכב עם מספר ממשי :

מספר-ממשי, שם-משתנה-מרוכב mult_comp_real

יתבצע חיבור של המשתנה המרוכב והמספר הממשי הנתונים בפקודה.

מספר-ממשי * מספר-מרוכב

תוצאת הפעולה תודפס לפלט בפורמט זהה לפורמט ההדפסה של סעיף 2.

6. כפל מספר מרוכב עם מספר מדומה :

מספר-מדומה, שם-משתנה-מרוכב mult_comp_img

יתבצע חיבור של המשתנה המרוכב והמספר המדומה הנתונים בפקודה.

המספר המדומה נתון בפקודה ללא הגורם i.

(i * מספר-מדומה) * מספר-מרוכב

תוצאת הפעולה תודפס לפלט בפורמט זהה לפורמט ההדפסה של סעיף 2.

7. כפל שני מספרים מרוכבים :

שם-משתנה-מרוכב-ב', שם-משתנה-מרוכב-א' mult_comp_comp

יתבצע כפל של שני המשתנים המרוכבים המופיעים בפקודה :

מספר-מרוכב-ב' * מספר-מרוכב-א'

תוצאת הפעולה תודפס לפלט בפורמט זהה לפורמט ההדפסה של סעיף 2.

8. ערך מוחלט של מספר מרוכב :

שם-משתנה-מרוכב abs_comp

יחושב ערכו המוחלט של המשתנה המרוכב שמופיע בפקודה :

| מספר-מרוכב |

תוצאת הפעולה תודפס לפלט בפורמט זהה לפורמט ההדפסה של סעיף 2.

9. סיום התכנית :

stop

פקודה זו היא ללא פרמטרים, ומטרתה לסיים את התכנית.

המבנה התחבירי של הקלט :

- כל פקודה תופיע בשלמותה בשורת קלט יחידה, כולל כל הפרמטרים. מותרות גם שורות ריקות (שורות המכילות רק תווים לבנים).

- שם הפקודה מופרד מהפרמטר הראשון באמצעות רווחים ו/או טאבים (אחד או יותר).
- בין כל שני אופרנדים יש פסיק אחד. לפני ואחרי הפסיק יכולים להיות רווחים ו/או טאבים בכמות בלתי מוגבלת. אסור שיהיה פסיק אחרי הפרמטר האחרון.
- יכולים להיות רווחים ו/או טאבים בכמות בלתי מוגבלת לפני שם הפקודה, וגם בסוף השורה (אחרי הפרמטר האחרון).
- אסור שיהיו תווי זבל בסוף השורה (למעט תווי לבנים).
- שמות הפקודות יופיעו באותיות קטנות בלבד, ושמות המשתנים באותיות גדולות בלבד.

אופן פעולת התכנית:

יש לממש ממשק משתמש ידידותי, כך שהמשתמש יוכל להבין בכל שלב של התכנית מה עליו לעשות. בפרט, על התכנית להודיע באמצעות הודעה או סימן (prompt) בכל פעם שהיא מוכנה לקלוט את הפקודה הבאה. התכנית תמשיך לקלוט ולבצע פקודה אחרי פקודה, עד שתקבל הפקודה stop.

התכנית אינה מניחה שהקלט תקין. על התכנית לנתח כל פקודה ולוודא שאין בה שגיאות (ראו דוגמאות בהמשך). במידה ונתגלתה שגיאה, התכנית תדפיס הודעת שגיאה פרטנית, ותעבור לפקודה הבאה, בלי לבצע את הפקודה השגויה. אין לעצור את התכנית עם גילוי השגיאה הראשונה. אין צורך לדווח על יותר משגיאה אחת בכל שורת קלט.

יש לטפל גם במצב של EOF (גמר הקלט). עצירת התכנית שלא באמצעות פקודת stop אינה נחשבת תקינה (גם לא כאשר הקלט מגיע מקובץ באמצעות redirection), ויש להדפיס הודעת שגיאה על כך ואז לעצור. שימו לב: השורה האחרונה בקובץ קלט אינה חייבת להסתיים בתו '\n'.

להלן דוגמאות של קלט שגוי:

שימו לב: ייתכנו סוגים נוספים של שגיאות בקלט. עליכם לחשוב על כל מגוון השגיאות האפשריות, ולטפל בכולן.

1. לפקודה: `read_comp G, 3.1, 6.5`
Undefined complex variable
יש להגיב בהודעה כגון:
2. לפקודה: `read_comp a, 3.6, 5.1`
Undefined complex variable
יש להגיב בהודעה כגון:
3. לפקודה: `do_it A, B`
Undefined command name
יש להגיב בהודעה כגון:
4. לפקודה: `Add_Comp A, C`
Undefined command name
יש להגיב בהודעה כגון:
5. לפקודה: `read_comp A, 3.5, xyz`
Invalid parameter – not a number
יש להגיב בהודעה כגון:
6. לפקודה: `read_comp A, 3.5`
Missing parameter
יש להגיב בהודעה כגון:
7. לפקודה: `read_comp A, 3.5, -3,`

Extraneous text after end of command	יש להגיב בהודעה כגון :
add_comp B	8. לפקודה :
Missing parameter	יש להגיב בהודעה כגון :
print_comp C, D	9. לפקודה :
Extraneous text after end of command	יש להגיב בהודעה כגון :
sub_comp F, , D	10. לפקודה :
Multiple consecutive commas	יש להגיב בהודעה כגון :
mult_comp_comp F D	11. לפקודה :
Missing comma	יש להגיב בהודעה כגון :
mult_comp_real, A, 2.5	12. לפקודה :
Illegal comma	יש להגיב בהודעה כגון :
mult_comp_img A, B	13. לפקודה :
Invalid parameter – not a number	יש להגיב בהודעה כגון :
abs_comp	14. לפקודה :
Missing parameter	יש להגיב בהודעה כגון :
abs_comp 2.5	15. לפקודה :
Undefined complex variable	יש להגיב בהודעה כגון :
stop A	16. לפקודה :
Extraneous text after end of command	יש להגיב בהודעה כגון :

להלן דוגמה של סדרת פקודות שכולן תקינות :

הערה : סדרה כגון זו יכולה לשמש כקלט בהרצת בדיקה של התכנית על קלט תקין.

```

print_comp A
print_comp B
print_comp C
read_comp A, 45.1, -23.75
print_comp A
read_comp B, 54.2, 3.56
print_comp B
read_comp C, 0, -1
print_comp C
add_comp A, B
sub_comp C, A
sub_comp B, B

```



```

sub_comp D, A
mult_comp_real A, 2.51
mult_comp_img A, -2.564
mult_comp_comp A, B
mult_comp_comp E , C
abs_comp A
abs_comp B
abs_comp C
abs_comp F
stop

```

דרישות נוספות :

- יש לחלק את התוכנית למספר קבצי מקור : complex.c , mycomp.c , ו-complex.h.
 - בקובץ complex.c יש לרכז את הפעולות החשבוניות על מספרים מרוכבים. לכל פעולה יש לממש פונקציה נפרדת, עם פרמטרים לפי מפרט הפעולה המוגדר לעיל.
 - בקובץ mycomp.c תהיה התכנית הראשית main, וכן כל פעילויות האינטראקציה עם המשתמש וניתוח הקלט (לרבות הדפסת הודעות השגיאה). כמו כן, יוגדרו בקובץ זה ששת המשתנים מטיפוס complex.
 - בקובץ complex.h תהיה הגדרת טיפוס הנתונים complex, וכן ההצהרות (אב-טיפוס) של הפונקציות הממומשות בקובץ complex.c. יש לכלול (#include) את הקובץ complex.h בקבצי המקור האחרים.
 - באפשרותכם לבנות קבצי מקור נוספים (למשל : קובץ המכיל פונקציות עזר לניתוח הקלט, וכד').
- הקלט לתכנית הוא מ-stdin, ויכול להגיע מהמקלדת או מקובץ (באמצעות redirection בעת הפעלת התכנית). לנוחיותכם, הכינו מספר קבצי קלט והשתמשו בהם שוב ושוב לדיבוג התכנית. בכל קובץ קלט תהיה סדרה של פקודות על מספרים מרוכבים.
- לפני הניתוח של כל שורת קלט, על התכנית להדפיס את השורה לפלט, בדיוק כפי שנקראה. זאת כדי שניתן יהיה לראות בפלט את הפקודות המקוריות, גם כאשר הקלט מגיע מקובץ.
- חובה לצרף להגשה הרצות בדיקה (אחת או יותר), המדגימות את השימוש בכל סוגי הפקודות ובכל ששת המשתנים המרוכבים, וכן את הטיפול בכל מגוון השגיאות בקלט. יש להגיש קובץ קלט + תדפיס מסך (או קובץ פלט) של כל הרצה.

להזכירכם : לא תנתן דחייה בהגשת הממ"ן, פרט למקרים מיוחדים כגון מילואים או מחלה, במקרים אלו יש לקבל אישור הגשה מצוות הקורס.

בהצלחה !

מטלת מנחה (ממ"ן) 23

הקורס: 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטלה: פרקים 6,7,8

מספר השאלות: 2 משקל המטלה: 12 נקודות (רשות)

סמסטר: 2019א' מועד אחרון להגשה: 6.1.2019

קיימות שתי חלופות להגשת מטלות:

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

יש לקמפל עם דגלים מקסימליים, לקבלת כל האזהרות: Wall-ansi-pedantic. יש להגיש את קבצי המקור (.c, .h), קבצי ההרצה (את קבצי .o אין צורך לצרף), קבצי הסביבה המתאימים (כולל קבצי makefile), וכן קבצי קלט ותדפיסי מסך או קבצי פלט (לפי הנחיות במטלה/במפגש/באתר). כל תכנית תהיה בתיקיה נפרדת. נדרש ששם התיקיה ושם הקובץ לריצה יהיו כשם הקובץ המכיל את הפונקציה main, ללא הסיומת .c. יש להגיש תכניות מלאות (בין השאר מכילות main), ניתנות להידור והרצה, ומאפשרות בדיקה של כל התוצאות המגוונות של הריצה ללא צורך בשינויים כלשהם בקוד התכנית. את המטלה יש להגיש בקובץ zip. לאחר ההגשה יש להוריד את המטלה משרת האו"פ למחשב האישי, ולבדוק שהקבצים אכן הוגשו באופן תקין.

שאלה 1 (10 נקודות)

עליכם להסביר מה מבצע קטע הקוד הבא, כאשר A ו-B הם משתנים מסוג int:

$A \wedge B$;

$B \wedge A$;

$A \wedge B$;

יש לנמק בדייקנות. השתמשו גם בדוגמה מספרית, לצורך הבהרה. הגישו את תשובתכם בקובץ מסוג word או pdf.

שאלה 2 (90 נקודות) (תכנית ראשית בקובץ timediff.c)

- א. (10 נקודות) עליכם להגדיר טיפוס נתונים בשם `time` המשמש לייצוג זמן. המידע המאוחסן הוא :
תאריך (לפי הלוח הגרגוריאני) : שנה, חודש, יום.
שעת היום : שעה (לפי 24 שעות), דקה, שניה.
- ב. (80 נקודות) עליכם לכתוב פונקציה בשם `time_diff` המקבלת שני פרמטרים מטיפוס `time`, ומחזירה את ההפרש בין שני הזמנים ביחידות של שניות. על הערך המוחזר להיות תמיד חיובי, אם כי לא ניתן להניח שיש סדר קבוע בין הזמנים של שני הפרמטרים. כלומר, על הפונקציה לבדוק מי משני הפרמטרים הוא הזמן המוקדם יותר ומי המאוחר יותר.
- כמו כן, עליכם לכתוב תכנית ראשית (הפונקציה `main`), אשר תגדיר ותאתחל שני משתנים מטיפוס `time`, ותקרא לפונקציה `time_diff`. התכנית הראשית תדפיס לפלט בפורמט נאה ומפורט את שני הזמנים (תאריך ושעת היום), ואת ההפרש ביניהם (בשניות).
- יש להפעיל את הפונקציה `time_diff` מספר פעמים באותה הרצה, עם ערכים מגוונים של זמנים, כדי להדגים את המקרים השונים של עבודת הפונקציה.

הנחיות נוספות :

- לשם פשטות, מותר להניח כי בחודש פברואר יש תמיד 28 ימים.
- שנים שלפני הספירה יצוינו באמצעות מספרים שליליים.
- ניתן להניח שכל ערך שיחושב לא יגלוש מהתחום המוגדר של הטיפוס `int`.
- אין להשתמש בפונקציות של הספריה הסטנדרטית המטפלות בזמן, כגון `time` או `diffime`.
- אין לבצע קלט מהמשתמש. כל הנתונים יוכנו בתכנית הראשית, ויהיו תמיד תקינים.

חובה לצרף להגשה הרצת בדיקה המדגימה את פעולת התכנית על נתונים מגוונים. יש להגיש **תדפיס מסך (או קובץ פלט)** של ההרצה.

להזכירכם : לא תנתן דחייה בהגשת הממ"ן, פרט למקרים מיוחדים כגון מילואים או מחלה, במקרים אלו יש לקבל אישור הגשה מצוות הקורס.

בהצלחה!

מטלת מנחה (ממ"ן) 14

הקורס: 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטלה: פרויקט גמר

מספר השאלות: 1 משקל המטלה: 31 נקודות

סמסטר: 2019א' מועד אחרון להגשה: 10.3.2019

קיימות שתי חלופות להגשת מטלות:

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

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

בפרויקט זה עליכם לכתוב תוכנת אסמבלר, עבור שפת אסמבלי שתוגדר בהמשך. הפרויקט ייכתב בשפת C.

עליכם להגיש:

1. קבצי המקור של התוכנית שכתבתם (קבצים בעלי סיומת c או h).
 2. קובץ הרצה (מקומפל ומקושר) עבור מערכת אונוטו.
 3. קובץ makefile. יש להשתמש בקומפיילר gcc עם הדגלים: -Wall -ansi -pedantic ולנפות את ההודעות שמוציא הקומפיילר, כך שהתכנית תתקמפל ללא הערות או אזהרות.
 4. דוגמאות הרצה (קלט ופלט):
- א. קבצי קלט בשפת אסמבלי, וקבצי הפלט שנוצרו מהפעלת האסמבלר על קבצי קלט אלה. יש להדגים שימוש במגוון הפעולות וטיפוסי הנתונים של שפת האסמבלי.
- ב. קבצי קלט בשפת אסמבלי המדגימים מגוון רחב של סוגי שגיאות אסמבלי (ולכן לא נוצרים קבצי פלט), ותדפיסי המסך המראים את הודעות השגיאה שמוציא האסמבלר.

בשל גודל הפרויקט, עליכם לחלק את התוכנית למספר קבצי מקור, לפי משימות. יש להקפיד שקוד התכנית יעמוד בקריטריונים של בהירות, קריאות וכתובה נאה ומובנית.

נזכיר מספר היבטים חשובים:

1. הפשטה של מבני הנתונים: רצוי (במידת האפשר) להפריד בין הגישה למבני הנתונים לבין המימוש של מבני הנתונים. כך, למשל, בכתיבת שגרות לטיפול בטבלה, אין זה מעניינם של המשתמשים בשגרות אלה, האם הטבלה ממומשת באמצעות מערך או באמצעות רשימה מקושרת.
2. קריאות הקוד: יש להשתמש בשמות משמעותיים למשתנים ופונקציות. כמו כן, רצוי להגדיר קבועים רלוונטיים תוך שימוש בפקודת #define, ולהימנע משימוש ב-"מספרי קסם", שמשמעותם נהירה לכם בלבד. יש לערוך את הקוד באופן מסודר: הזחות עקביות, שורות ריקות להפרדה בין קטעי קוד, וכד'.
3. תיעוד: יש להכניס בקבצי המקור תיעוד תמציתי וברור, שיסביר את תפקידה של כל פונקציה ופונקציה. כמו כן יש להסביר את תפקידים של משתנים חשובים. כמו כן, יש להכניס הערות הסבר ברמת פירוט טובה בכל הקוד.

הערה: תוכנית "עובדת", דהיינו תוכנית שמבצעת את הדרוש ממנה, אינה ערובה לציון גבוה. כדי לקבל ציון גבוה על התכנית לעמוד בקריטריונים לעיל, אשר משקלם המשותף מגיע עד לכ-40% ממשקל הפרויקט.

מותר להשתמש בפרויקט בכל מגוון הספריות הסטנדרטיות של שפת C, אבל אין להשתמש בספריות חיצוניות אחרות.

מומלץ לעבוד בזוגות. אין לעבוד בצוותים גדולים יותר. **פרויקט שיוגש על ידי שלשה או יותר, לא ייבדק 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 ברצף, לקבלת תמונה כללית לגבי הנדרש, ורק לאחר מכן לקרוא בשנית, בצורה מעמיקה יותר.

רקע כללי ומטרת הפרויקט

כידוע, קיימות שפות תכנות רבות, ומספר גדול של תוכניות, הכתובות בשפות שונות, עשויות לרוץ באותו מחשב עצמו. כיצד "מכיר" המחשב כל כך הרבה שפות? התשובה פשוטה: המחשב מכיר למעשה שפה אחת בלבד: הוראות ונתונים הכתובים בקוד בינארי. קוד זה מאוחסן בגוש בזיכרון, ונראה כמו רצף של ספרות בינאריות. יחידת העיבוד המרכזית - היע"מ (CPU) - יודעת לפרק את הרצף הזה לקטעים קטנים בעלי משמעות: הוראות, מענים ונתונים. אופן הפירוק נקבע, באופן חד משמעי, על ידי המיקרו קוד של המעבד.

למעשה, זיכרון המחשב כולו הוא אוסף של סיביות, שנוהגים לראותן כמקובצות ליחידות בעלות אורך קבוע (בתים, מילים). לא ניתן להבחין, בעין שאינה מיומנת, בהבדל פיסי כלשהו בין אותן חלק בזיכרון שבו נמצאת תוכנית לבין שאר הזיכרון.

יחידת העיבוד המרכזית (היע"מ) יכולה לבצע מספר מסוים של הוראות פשוטות, ולשם כך היא משתמשת בזיכרון המחשב ובאוגרים (registers) הקיימים בתוך היע"מ. דוגמאות: העברת מספר מתא בזיכרון לאוגר ביע"מ או בחזרה, הוספת 1 למספר הנמצא באוגר, בדיקה האם מספר המאוחסן באוגר שווה לאפס. הוראות פשוטות אלה ושילובים שלהן הן המרכיבות את תוכנית המשתמש כפי שהיא נמצאת בזיכרון. כל תוכנית מקור (התוכנית כפי שנכתבה בידי המתכנת), נתורגם בסופו של דבר באמצעות תוכנה מיוחדת לצורה סופית זו.

קוד בשפת מכונה הוא רצף של ביטים המהווים קידוד של סדרת הוראות (תוכנית) שעל היע"מ לבצע. קוד מכונה אינו קריא למשתמש, ולכן לא נוח לקודד(או לקרוא) תכניות ישירות בשפת מכונה. שפת אסמבלי (assembly language) היא שפת תכנות מאפשרת לייצג את ההוראות של שפת המכונה בצורה סימבולית. כמובן שיש צורך לתרגם את הייצוג הסימבולי לקוד מכונה כדי שהתוכנית תוכל לרוץ במחשב. תרגום זה נעשה באמצעות כלי שנקרא אסמבלר (assembler).

כידוע, לכל שפת תכנות עילית יש מהדר (compiler), או מפרש (interpreter), המתרגם תוכניות מקור לשפת מכונה. האסמבלר משמש בתפקיד דומה עבור שפת אסמבלי.

לכל מודל של יע"מ (אירגון של מחשב) יש שפת מכונה יעודית משלו, ובהתאם גם שפת אסמבלי יעודית משלו. לפיכך, גם האסמבלר (כלי התרגום) הוא יעודי ושונה לכל יע"מ.

תפקידו של האסמבלר הוא לייצר קוד מכונה גולמי עבור קובץ של תכנית הכתובה בשפת אסמבלי. זהו השלב הראשון במסלול אותו עוברת התכנית, עד לקבלת קוד המוכן לריצה על גבי חומרת המחשב. השלבים הבאים הם קישור (linkage) וטעינה (loading), אך בהם לא נעסוק בממ"ן זה.

המשימה בפרויקט זה היא לכתוב אסמבלר (כלומר תוכנית המתרגמת לשפת מכונה), עבור שפת אסמבלי שנגדיר כאן במיוחד לצורך הפרויקט.

לתשומת לב: בהסברים הכלליים על אופן עבודת תוכנת האסמבלר, תהיה מדי פעם התייחסות גם לעבודת שלבי הקישור והטעינה. התייחסויות אילו נועדו על מנת לאפשר לכם להבין את המשך תהליך העיבוד של הפלט של תוכנת האסמבלר. אין לטעות: עליכם לכתוב את תוכנית האסמבלר בלבד, **אין** צורך לכתוב גם את תוכניות הקישור והטעינה!!!

המחשב הדמיוני ושפת האסמבלי

נגדיר עתה את שפת האסמבלי ואת מודל המחשב הדמיוני, עבור פרויקט זה.

הערה: תאור מודל המחשב להלן הוא חלקי בלבד, ככל שנחוץ לביצוע המשימות בפרויקט.

"חומרה":

המחשב בפרויקט מורכב מיע"מ (יחידת עיבוד מרכזית), אוגרים וזיכרון RAM, כאשר חלק מהזיכרון משמש גם כמחסנית (stack). גודלה של מילת זיכרון במחשב הוא 12 סיביות. האריתמטיקה נעשית בשיטת המשלים ל-2 (2^2 's complement). מחשב זה עובד רק עם מספרים שלמים חיוביים ושליילים, אין תמיכה במספרים ממשיים.

אוגרים:

למעבד 8 אוגרים כלליים, בשמות r0, r1, r2, r3, r4, r5, r6, r7. גודלו של כל אוגר הוא 12 סיביות. הסיבית הכי פחות משמעותית תצוין כסיבית מס' 0, והסיבית המשמעותית ביותר כמס' 11. שמות האוגרים נכתבים תמיד עם אות 'r' קטנה.

כמו כן יש במעבד אוגר בשם PSW (program status word), המכיל מספר דגלים המאפיינים את מצב הפעילות במעבד בכל רגע נתון. ראו בהמשך, בתאור הפקודות, לגבי השימוש בדגלים אלו.

גודל הזיכרון הוא 1024 תאים, בכתובות 0-1023 (בבסיס עשרוני), וכל תא הוא בגודל של 12 סיביות. לתא בזיכרון נקרא גם בשם "מילה". הסיביות בכל מילה ממוספרות בדומה לאוגר, כמפורט לעיל.

קידוד של תווים (characters) נעשה בקוד ascii.

מבנה הוראת מכונה:

כל הוראת מכונה מקודדת למספר מילות זיכרון, החל ממילה אחת ועד למקסימום שלוש מילים, הכל בהתאם לשיטות המיעון בהן נעשה שימוש (ראו בהמשך). בכל סוגי ההוראות, המבנה של המילה הראשונה זהה. מבנה המילה הראשונה בהוראה הוא כדלהלן:

11	9	8	5	4	2	1	0
מיעון אופרנד מקור		opcode		מיעון אופרנד יעד		A,R,E	

סיביות 5-8 במילה הראשונה של ההוראה מהוות את קוד הפעולה (opcode). כל opcode מיוצג בשפת אסמבלי על ידי "שם פעולה". בשפה שלנו יש 16 קודי פעולה והם:

שם הפעולה	קוד הפעולה (בבסיס 10)
mov	0
cmp	1
add	2
sub	3
not	4
clr	5
lea	6
inc	7
dec	8

שם הפעולה	קוד הפעולה (בבסיס 10)
jmp	9
bne	10
red	11
prn	12
jsr	13
rts	14
stop	15

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

סיביות 0-1 (A,R,E)

סיביות אלה מציינות את סוג הקידוד, האם הוא מוחלט (Absolute), חיצוני (External) או מצריך מיקום מחדש (Relocatable).

ערך של 00 משמעו שהקידוד הוא מוחלט.
ערך של 01 משמעו שהקידוד הוא חיצוני.
ערך של 10 משמעו שהקידוד מצריך מיקום מחדש.

סיביות אלה מתווספות רק לקידודים של הוראות (לא של נתונים), והן מתווספות גם לכל המילים הנוספות שיש לקידודים אלה.

ראו הסבר נוסף על סיביות אלה בהמשך.

סיביות 2-4 מקודדות את מספרה של שיטת המיעון של אופרנד היעד (destination operand).

סיביות 9-11 מקודדות את מספרה של שיטת המיעון של אופרנד המקור (source operand).

שיטות מיעון:

בשפה שלנו קיימות שלוש שיטות מיעון, שמסומנות במספרים 1, 3, 5.

השימוש בשיטות מיעון מצריך קידוד של מילות-מידע נוספות בקוד המכונה של ההוראה. אם בפקודה יש אופרנד אחד, תהיה מילת מידע אחת נוספת. אם בפקודה יש שני אופרנדים, ייתכנו שתי מילות-מידע נוספות, או מילת-מידע אחת המשותפת לשני האופרנדים, תלוי בשיטות המיעון בהן נעשה שימוש (ראו מפרט בהמשך). כאשר בקידוד הפקודה יש שתי מילות-מידע נוספות, אזי מילת-המידע הראשונה מתייחסת לאופרנד המקור, והשנייה מתייחסת לאופרנד היעד.

בכל מילת-מידע נוספת, סיביות 0-1 הן השדה A,R,E.

להלן תיאור של שיטות המיעון :

קוד	שיטת מיעון	תוכן המילים הנוספות	אופן הכתיבה	דוגמא
1	מיעון מידי	מילת-מידע נוספת של ההוראה מכילה את האופרנד עצמו, שהוא מספר המיוצג ב- 10 סיביות, אליהן מתווספות זוג סיביות של השדה A,R,E.	האופרנד הוא מספר שלם בבסיס עשרוני	mov -103,@r2 בדוגמה זו האופרנד הראשון של הפקודה נתון בשיטת מיעון מיידי. ההוראה כותבת את הערך 103- אל אוגר r2
3	מיעון ישיר	מילת-מידע נוספת של ההוראה מכילה כתובת של מילה בזיכרון. מילה זו בזיכרון היא האופרנד. המען מיוצג ב- 10 סיביות אליהן מתווספות זוג סיביות של השדה A,R,E.	האופרנד הוא <u>תווית</u> שכבר הוצהרה או שתוצהר בהמשך הקובץ. ההצהרה נעשית על ידי כתיבת תווית בתחילת הנחיית 'data' או 'string', או בתחילת הוראה של התוכנית, או באמצעות אופרנד של הנחיית 'extern'.	נתונה למשל ההגדרה : x: .data 23 אפשר לכתוב הוראה : dec x בדוגמה זו, ההוראה מקטינה ב-1 את תוכן המילה שבכתובת x בזיכרון (ה"משתנה" x).
5	מיעון אוגר ישיר	האופרנד הוא אוגר. אם האוגר משמש כאופרנד יעד, מילת-מידע נוספת של הפקודה תקודד בחמש הסיביות 2-6 את מספרו של האוגר. ואילו אם האוגר משמש כאופרנד מקור, מספר האוגר יקודד בחמש הסיביות 7-11 של מילת-המידע. אם בפקודה יש שני אופרנדים, ושניהם אוגרים, הם יחלקו מילת-מידע אחת משותפת, כאשר הסיביות 2-6 הן עבור אוגר היעד, והסיביות 7-11 עבור אוגר המקור. בכל מילה נוספת מתווספות זוג סיביות של השדה A,R,E. סיביות שאינן בשימוש יכילו 0.	האופרנד הוא שם של אוגר. שם האוגר ייכתב כאשר לפניו יופיע התו @	mov @r1,@r2 בדוגמה זו, ההוראה מעתיקה את תוכן אוגר r1 אל אוגר r2. בדוגמה זו שני האופרנדים הם אוגרים, אשר יקודדו למילת-מידע נוספת אחת משותפת.

הערה: מותר להתייחס לתווית עוד לפני שמצהירים עליה, בתנאי שהיא אכן מוצהרת במקום כלשהו בקובץ.

אפיון הוראות המכונה :

הוראות המכונה מתחלקות לשלוש קבוצות, לפי מספר האופרנדים הדרוש להן.

קבוצת ההוראות הראשונה :

הוראות הדורשות שני אופרנדים. ההוראות השייכות לקבוצה זו הן :

mov, cmp, add, sub, lea

הפעולה	הסבר הפעולה	דוגמא	הסבר הדוגמא
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 בזכרון) אל אוגר r1.
cmp	מבצעת "השוואה" בין שני האופרנדים שלה. אופן ההשוואה : תוכן אופרנד היעד (השני) מופחת מתוכן אופרנד המקור (הראשון), ללא שמירת תוצאת החיסור. פעולת החיסור מעדכנת דגל בשם Z ("דגל האפס") באוגר הסטטוס (PSW).	cmp A, @r1	אם תוכן המשתנה A זהה לתוכנו של אוגר r1 אזי דגל האפס, Z, באוגר הסטטוס (PSW) יודלק, אחרת הדגל יאופס.
add	אופרנד היעד (השני) מקבל את תוצאת החיבור של אופרנד המקור (הראשון) והיעד (השני).	add A,@r0	אוגר r0 מקבל את תוצאת החיבור של תוכן המשתנה A ותוכנו הנוכחי של אוגר r0.
sub	אופרנד היעד (השני) מקבל את תוצאת החיסור של אופרנד המקור (הראשון) מאופרנד היעד (השני).	sub 3,@r1	אוגר r1 מקבל את תוצאת החיסור של הערך 3 מתוכנו הנוכחי של האוגר r1.
lea	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 לאוגר r1.

קבוצת ההוראות השנייה :

אלו הוראות הדורשות אופרנד אחד בלבד. אופן הקידוד של האופרנד הוא כמו של אופרנד היעד בפקודה עם שני אופרנדים. במקרה זה, השדה של אופרנד המקור (סיביות 4-2) במילה הראשונה בקידוד ההוראה הינו חסר משמעות, ולפיכך יכול 0. על קבוצה זו נמנות ההוראות הבאות :

not, clr, inc, dec, jmp, bne, red, prn, jsr

פקודה	הסבר פעולה	דוגמא	הסבר דוגמא
not	היפוך ערכי הסיביות באופרנד (כל סיבית שערכה 0 תהפוך ל-1 ולהיפך : 1 ל-0).	not @r2	$r2 \leftarrow \text{not } r2$
clr	איפוס תוכן האופרנד.	clr @r2	$r2 \leftarrow 0$
inc	הגדלת תוכן האופרנד באחד.	inc @r2	$r2 \leftarrow r2 + 1$
dec	הקטנת תוכן האופרנד באחד.	dec C	$C \leftarrow C - 1$

פקודה	הסבר פעולה	דוגמא	הסבר דוגמא
jmp	קפיצה (הסתעפות) בלתי מותנית אל ההוראה שנמצאת במען המיוצג על ידי האופרנד. כלומר, בעת ביצוע ההוראה, מצביע התוכנית (PC) יקבל את ערך האופרנד היעד.	jmp LINE	PC ← LINE מצביע התוכנית מקבל את המען המיוצג על ידי התווית LINE, ולפיכך הפקודה הבאה שתבצע תהיה במען זה.
bne	bne הוא קיצור (ראשי תיבות) של branch if not equal (to zero). זוהי הוראת ההסתעפות מותנית. מצביע התוכנית (PC) יקבל את ערך האופרנד היעד אם ערכו של הדגל Z באוגר הסטטוס (PSW) הינו 0. כזכור, ערך הדגל Z נקבע בהוראת cmp.	bne LINE	אם ערך הדגל Z באוגר הסטטוס (PSW) הינו 0: PC ← LINE
red	קריאה של תו מהקלט הסטנדרטי (stdin) אל האופרנד.	red @r1	קוד ה-ascii של התו הנקרא מהקלט הסטנדרטי ייכנס לאוגר r1.
prn	הדפסת התו הנמצא באופרנד, אל הפלט הסטנדרטי (stdout).	prn @r1	התו אשר קוד ה-ascii שלו נמצא באוגר r1 יודפס לפלט הסטנדרטי.
jsr	קריאה לשגרה (סברוטניה), מצביע התוכנית (PC) הנוכחי נדחף לתוך המחסנית שבזכרון המחשב, והאופרנד מוכנס ל-PC.	jsr FUNC	push(PC) PC ← FUNC

קבוצת ההוראות השלישית:

אלו הוראות ללא אופרנדים. קידוד ההוראה מורכב ממילה אחת בלבד. במקרה זה, השדות של אופרנד המקור ואופרנד היעד אינם רלוונטים (כי אין אופרנדים), ויכילו 0.

ההוראות השייכות לקבוצה זו הן: stop, rts.

פקודה	הסבר פעולה	דוגמא	הסבר דוגמא
rts	חזרה משיגרה. הערך בראש המחסנית של זמן-ריצה מוצא מן המחסנית ומוכנס אל מצביע התוכנית (PC).	rts	PC ← pop()
stop	עצירת ריצת התוכנית.	stop	התכנית עוצרת

מספר נקודות נוספות לגבי תיאור שפת האסמבלי:

שפת האסמבלי מורכבת ממשפטים (statements) כאשר התו המפריד בין משפט למשפט הינו תו 'מ' (שורה חדשה). כלומר, כאשר מסתכלים על קובץ המקור, רואים אותו כמורכב משורות של משפטים, כאשר כל משפט מופיע בשורה נפרדת.

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

ישנם ארבעה סוגי משפטים (שורות) בשפת האסמבלי, והם:

סוג המשפט	הסבר כללי
משפט ריק	זוהי שורה המכילה אך ורק תווים לבנים (whitespace), כלומר מכילה רק תווים 't' ו-' ' (סימני tab ורווח). ייתכן ובשורה אין אף תו (למעט התו מ).
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 להקצאה ואתחול משתנים של התכנית, אך הוא אינו מייצר קוד המיועד לביצוע בעת ריצת התכנית.
משפט הוראה	זהו משפט המייצר קוד לביצוע בעת ריצת התכנית. המשפט מכיל שם של פעולה שעל ה-CPU לבצע, ותיאור האופרנדים המשתתפים בפעולה.

כעת נפרט לגבי סוגי המשפטים השונים.

משפט הנחיה:

משפט הנחיה הוא בעל המבנה הבא:

בתחילתו יכולה להופיע תווית (label) (התווית חייבת להיות בתחביר חוקי. התחביר של תווית חוקית יתואר בהמשך). התווית היא אופציונלית.

לאחר מכן מופיע התו ' (נקודה), ובצמוד לנקודה (ללא רווח) שם ההנחיה, באותיות קטנות (lower case) בלבד. לאחר שם ההנחיה יופיעו פרמטרים (מספר הפרמטרים נקבע בהתאם לסוג ההנחיה).

יש לשים לב: למילות הקוד הנוצרות ממשפט הנחיה לא מצורפות זוג סיביות A,R,E והקידוד ממלא את כל 12 הסיביות של המילה.

יש ארבעה סוגים של משפטי הנחיה, והם:

1. 'data'.

הפרמטר(ים) של data הם רשימת מספרים שלמים חוקיים (אחד או יותר) המופרדים על ידי התו ' (פסיק). למשל:

data 7, -57, +17, 9.

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

משפט ההנחיה: 'data'. מנחה את האסמבלר להקצות מקום בתמונת הנתונים (data image), אשר בו יאוחסנו הערכים המתאימים, ולקדם את מונה הנתונים, בהתאם למספר הערכים ברשימה. אם להוראת data יש גם תווית, אזי תווית זו מקבלת את ערך מונה הנתונים (לפני הקידום), ומוכנסת אל טבלת הסמלים. דבר זה מאפשר להתייחס אל מקום מסוים בתמונת הנתונים דרך שם התווית (למעשה, זוהי הגדרת שם של משתנה).

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 הוראה לביצוע:

```
mov XYZ, @r1
```

אזי בזמן ריצת התכנית יוכנס לאוגר r1 הערך 7.

ואילו ההוראה:

```
lea XYZ, @r1
```

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

2. 'string'.

להנחיה 'string' פרמטר אחד שהוא מחרוזת חוקית. תווי המחרוזת מקודדים לפי ערכי ה-ascii המתאימים ומוכנסים אל תמונת הנתונים, לפי סדרם, כל תו במילה נפרדת. בסוף המחרוזת יוכנס הערך אפס, לסמן סיום מחרוזת. מונה הנתונים של האסמבלר יוגדל בהתאם לאורך המחרוזת + 1. אם בשורת ההנחיה מגדרת גם תווית, אזי ערכה יהיה מען המקום בזיכרון, שבו מאוחסן התו הראשון במחרוזת, באופן דומה כפי שנעשה עבור 'data'.

לדוגמא, משפט ההנחיה:

```
STR: .string "abcdef"
```

מקצה בתמונת הנתונים "מעריך תווים" באורך של 7 מקומות החל מהמען המזוהה עם התווית ABC, ומאתחל "מעריך" זה לערכי ה-ascii של התווים: a, b, c, d, e, f בהתאמה, ולאחריהם הערך 0 לסימון סוף מחרוזת. התווית STR מזוהה עם כתובת התחלת המחרוזת.

3. 'entry'.

להנחיה 'entry' פרמטר אחד והוא שם של תווית המוגדרת בקובץ המקור הנוכחי (כלומר מקבלת את ערכה בקובץ זה). מטרת entry היא להצהיר על התווית הזו באופן שיאפשר לקטעי אסמבלי הנמצאים בקבצי מקור אחרים להשתמש בה.

לדוגמא, השורות:

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

מודיעות שאפשר להתייחס מקובץ 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).

4. 'extern'.

להנחיה 'extern' פרמטר אחד והוא שם של תווית. מטרת ההוראה היא להצהיר כי התווית מוגדרת בקובץ אחר וכי קטע האסמבלי, בקובץ זה, עושה בו שימוש. זוהי הנחיה תואמת להנחיית entry המופיעה בקובץ בו מוגדרת התווית. בזמן הקישור (link) תתבצע ההתאמה, בין ערך התווית, כפי שנקבע בקוד המכונה של הקובץ שהגדיר את התווית, לבין קוד ההוראות המשתמשות בו בקבצים אחרים.

לדוגמא, משפט ההנחיה 'extern' התואם למשפט ההנחיה 'entry' בדוגמא הקודמת יהיה:

HELLO extern.

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

משפט הוראה:

משפט הוראה מורכב מ:

1. תווית אופציונלית.
2. שם פעולה.
3. 0, 1 או 2 אופרנדים בהתאם לסוג הפעולה.

שם הפעולה נכתב תמיד באותיות קטנות (lower case), והפעולה היא אחת מבין 16 הפעולות שהוזכרו לעיל.

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

כאשר יש שני אופרנדים, האופרנדים מופרדים זה מזה בתו ' ', (פסיק). כמו בהנחה data, לא חייבת להיות שום הצמדה של האופרנדים לפסיק או לשם הפעולה באופן כלשהו. כל כמות רווחים או tabs בין האופרנדים לפסיק ובין שם הפעולה לאופרנד הראשון היא חוקית.

להוראה בעלת שני אופרנדים המבנה הבא:

אופרנד-יעד, אופרנד-מקור שם-הפעולה תווית-אופציונלית

לדוגמא:

HELLO: add @r7, B

לפקודה בעלת אופרנד אחד המבנה הבא:

אופרנד שם-הפעולה תווית-אופציונלית

לדוגמא:

HELLO: bne XYZ

להוראה ללא אופרנדים המבנה הבא:

שם-הפעולה תווית-אופציונלית

לדוגמא:

END: stop

אם מופיעה תווית בשורת ההוראה, אזי היא תוכנס אל טבלת הסמלים. ערך התווית יהיה מען המילה הראשונה של ההוראה בתוך תמונת הקוד שבונה האסמבלר.

תווית:

תווית חוקית מתחילה באות אלפבית (גדולה או קטנה) ולאחריה סדרה כלשהי של אותיות אלפבית (גדולות או קטנות) וספרות. האורך המקסימלי של תווית הוא 31 תווים. הגדרת התווית מסתיימת על ידי התו ': ' (נקודתיים). תו זה אינו מהווה חלק מהתווית. זהו רק סימן המציין את סופה. הגדרת התווית חייבת להתחיל בעמודה הראשונה בשורה. אסור שאותה תווית תוגדר יותר מפעם אחת (כמובן בשורות שונות). אותיות קטנות וגדולות נחשבות שונות זו מזו.

לדוגמה, התוויות המוגדרות להלן הן תוויות חוקיות.

hEllo:

x:

He78902:

לתשומת לב: מילים שמורות של שפת האסמבלי (כלומר שם של פעולה או הנחיה, או שם של אוגר) אינן יכולות לשמש כשם של תווית.

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

מספר:

מספר חוקי מתחיל בסימן אופציונלי '-' או '+' ולאחריו סדרה כלשהי של ספרות בבסיס עשר. הערך של המספר הוא הערך המיוצג על ידי מחרוזת הספרות והסימן. כך למשל, 123, -5, 76, הינם מספרים חוקיים. אין תמיכה בשלמים בייצוג אחר מאשר עשרוני, ואין תמיכה במספרים ממשיים.

מחרוזת:

מחרוזת חוקית היא סדרת תווי `ascii` נראים, המוקפים במרכאות כפולות (המרכאות אינן נחשבות כחלק מהמחרוזת). דוגמא למחרוזת חוקית: "hello world".

איפיון השדה A,R,E בקוד המכונה

בכל מילה בקוד המכונה של הוראה (לא של נתונים), האסמבלר מכניס מידע עבור תהליך הקישור והטעינה. זהו השדה `A,R,E` (שתי הסיביות הימניות 0-1). המידע ישמש לתיקונים בקוד בכל פעם שייטען לזיכרון לצורך הרצה. האסמבלר בונה מלכתחילה קוד שמיועד לטעינה החל מכתובת 100. התיקונים יאפשרו לטעון את הקוד בכל פעם למקום אחר, בלי צורך לחזור על תהליך האסמבלי.

שתי הסיביות בשדה `A,R,E` יכילו את אחד הערכים הבינאריים: 0, 10, או 01. 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). במקרה זה שתי הסיביות הימניות יכילו את הערך 00.

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). במקרה זה שתי הסיביות הימניות יכילו את הערך 10.

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). במקרה זה שתי הסיביות הימניות יכילו את הערך 01.

אסמבלר עם שני מעברים

כאשר מקבל האסמבלר קוד לתרגום, עליו לבצע שתי משימות עיקריות לצורך הכנת הקוד הבינארי: הראשונה היא לזהות ולתרגם את שמות הפעולות, והשנייה היא לקבוע מענים לכל הסמלים המופיעים בתוכנית.

לדוגמא: האסמבלר קורא את קטע הקוד הבא:

```

MAIN:      mov    @r3,LENGTH
LOOP:      jmp     L1
           prn     -5
           bne     LOOP
           sub     @r1,@r4
           bne     END
L1:         inc     K
           bne     LOOP
END:        stop
STR:        .string "abcdef"
LENGTH:     .data   6,-9,15
K:          .data   22

```

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

כמו כן, על האסמבלר להחליף את הסמלים K,STR, LENGTH, L1, MAIN, LOOP, END במענים של המקומות בזיכרון שם נמצאים כל נתון או הוראה בהתאמה.

נניח שקטע הקוד לעיל (הוראות ונתונים) ייטען בזיכרון החל ממען 100 (בבסיס 10). במקרה זה נקבל את ה"תרגום" הבא:

Decimal Address	Source Code	Explanation	Binary Machine Code
0100 0101 0102	MAIN: mov @r3,LENGTH	first word of instruction source register 3 address of label LENGTH	101000001100 000110000000 000111110110
0103 0104	LOOP: jmp L1	address of label L1	000100101100 000111000110
0105 0106	prn -5	immediate value -5	000110000100 111111101100
0107 0108	bne LOOP	address of label LOOP	000101001100 000110011110
0109 0110	sub @r1, @r4	registers r1 and r4	101001110100 000010010000
0111 0112	bne END	address of label END	000101001100 000111010110
0113 0114	L1: inc K	address of label K	000011101100 001000000010
0115 0116	bne LOOP	address of label LOOP	000101001100 000110011110
0117	END: stop		000111100000

Decimal Address	Source Code	Explanation	Binary Machine Code
0118 0119 0120 0121 0122 0123 0124	STR: .string "abcdef"	ascii code 'a' ascii code 'b' ascii code 'c' ascii code 'd' ascii code 'e' ascii code 'f' ascii code '\0' (end of string)	000001100001 000001100010 000001100011 000001100100 000001100101 000001100110 000000000000
0125 0126 0127	LENGTH: .data 6,-9,15	integer 6 integer -9 integer 15	000000000110 111111110111 000000001111
0128	K: .data 22	integer 22	000000010110

האסמבלר מחזיק טבלה שבה רשומים כל שמות הפעולה של ההוראות והקודים הבינאריים המתאימים להם, ולכן שמות הפעולות ניתנים להמרה לבינארי בקלות. כאשר נקרא שם פעולה, 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, הידועים מראש, הרי המענים בזיכרון עבור הסמלים שבשימוש התכנית אינו ידוע, עד אשר התוכנית כולה נקראת ועוברת טיפול על ידי האסמבלר.

למשל, בדוגמא לעיל, האסמבלר אינו יכול לדעת שהסמל END משויך למען 117 (עשרוני), אלא רק לאחר שהתוכנית נקראה כולה.

לכן מפרידים את הטיפול של האסמבלר בסמלים לשני שלבים. בשלב הראשון בונים טבלה של כל הסמלים והערכים 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 בזיכרון, ובה לכל סמל שבתוכנית המקור משויך ערך שהוא מען בזיכרון. בדוגמא דלעיל, טבלת הסמלים לאחר מעבר ראשון היא:

ערך (בבסיס דצימלי)	סמל
100	MAIN
103	LOOP
113	L1
117	END
118	STR
125	LENGTH
128	K

במעבר השני נעשית ההמרה, כדי לתרגם את קוד המקור לקוד מכונה. בתחילת המעבר השני צריכים הערכים של הסמלים להיות כבר ידועים.

לתשומת לב: האסמבלר, על שני המעברים שלו, נועד כדי לתרגם קובץ מקור לקוד בשפת מכונה. בגמר פעולת האסמבלר, התכנית טרם מוכנה לטעינה לזיכרון לצורך ביצוע. לאחר השלמת תהליך התרגום, קוד המכונה יכול לעבור לשלבי הקישור/טעינה ולאחר מכן לשלב הביצוע.

המעבר הראשון

במעבר הראשון נדרשים כללים כדי לקבוע איזה מען ישויך לכל סמל. העיקרון הבסיסי הוא לספור את המקומות בזיכרון, אותם תופסות ההוראות. אם כל הוראה תיטען בזיכרון למקום העוקב להוראה הקודמת, תציין ספירה כזאת את מען ההוראה הבאה. הספירה נעשית על ידי האסמבלר ומוחזקת במונה ההוראות (IC). ערכו ההתחלתי של IC הוא 0, ולכן נטענת ההוראה הראשונה במען 0. ה-IC מתעדכן בכל שורת הוראה המקצה מקום בזיכרון. לאחר שהאסמבלר

קובע מהו אורך ההוראה, ה-IC מוגדל במספר התאים (מילים) הנתפסים על ידי ההוראה, וכך הוא מצביע על התא הפנוי הבא.

כאמור, כדי לקודד את ההוראות בשפת מכונה, מחזיק האסמבלר טבלה, שיש בה קוד מתאים לכל שם פעולה. בזמן התרגום מחליף האסמבלר כל שם פעולה בקוד שלה, וכל אופרנד בקידוד מתאים. אך פעולת ההחלפה אינה כה פשוטה. ההוראות משתמשות בשיטות מיעון מגוונות לאופרנדים. אותה פעולה יכולה לקבל משמעויות שונות, בכל אחת משיטות המיעון, ולכן יתאימו לה קידודים שונים לפי שיטות המיעון. לדוגמא, פעולת ההזזה mov יכולה להתייחס להעתקת תוכן תא זיכרון לאוגר, או להעתקת תוכן אוגר לאוגר אחר, וכן הלאה. לכל אפשרות כזאת של mov עשוי להתאים קידוד שונה.

על האסמבלר לסרוק את שורת ההוראה בשלמותה, ולהחליט לגבי הקידוד לפי האופרנדים. בדרך כלל מתחלק הקידוד לשדה של שם הפעולה, ושדות נוספים המכילים מידע לגבי שיטות המיעון.

במחשב שלנו קיימת גמישות לגבי שיטת המיעון של כל אחד מהאופרנדים בנפרד. הערה: דבר זה לא מחייב לגבי כל מחשב. ישנם מחשבים בהם, למשל, כל הפקודות הן בעלות אופרנד יחיד (והפעולות מתבצעות על אופרנד זה ועל אוגר קבוע). יש גם מחשבים עם פקודות של שלשה אופרנדים (כאשר האופרנד השלישי משמש לאחסון תוצאת הפעולה), ועוד אפשרויות אחרות.

כאשר נתקל האסמבלר בתווית המופיעה בתחילת השורה, הוא יודע שלפניו הגדרה של תווית, ואז הוא משייך לה מען – תוכנו הנוכחי של ה-IC. כך מקבלות כל התוויות את מעניהן בעת ההגדרה. תוויות אלה מוכנסות לטבלת הסמלים, המכילה בנוסף לשם התווית גם את המען ומאפיינים נוספים. כאשר תהיה התייחסות לתווית באופרנד של הוראה כלשהי, יוכל האסמבלר לשלוף את המען המתאים מטבלת הסמלים.

הוראה יכולה להתייחס גם לסמל שטרם הוגדר עד כה בתכנית, אלא יוגדר רק בהמשך התכנית. להלן לדוגמא, הוראת הסתעפות למען שמוגדר על ידי התווית A שמופיעה רק בהמשך הקוד:

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

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

בכל מקרה, תמיד אפשר לייצר במעבר הראשון את הקוד הבינארי המלא של המילה הראשונה של כל פקודה, וכן את הקוד הבינארי של כל הנתונים (המתקבלים מההנחיות .data, .string).

המעבר השני

במעבר הראשון, האסמבלר אינו יכול לבנות את קוד המכונה של אופרנדים המשתמשים בסמלים שלא הוגדרו עדיין. רק לאחר שהאסמבלר עבר על כל התכנית, כך שכל התוויות נכנסו כבר לטבלת הסמלים, יכול האסמבלר להשלים את קוד המכונה של כל האופרנדים. לשם כך עובר האסמבלר שנית על כל התוכנית, ומחליף את התוויות, המופיעות באופרנדים, במעניהן המתאימים מתוך הטבלה. זהו המעבר השני, ובסופו תהיה התוכנית מתורגמת בשלמותה.

הפרדת הוראות ונתונים

בתכנית מבחינים בשני סוגים של תוכן: הוראות ונתונים. יש לארגן את קוד המכונה כך שתהיה הפרדה בין הנתונים וההוראות. הפרדת ההוראות והנתונים לקטעים שונים בזיכרון היא שיטה עדיפה על פני הצמדה של הגדרות הנתונים להוראות המשתמשות בהן.

אחת הסכנות הטמונות באי הפרדת ההוראות מהנתונים היא, שלפעמים עלול המעבד, בעקבות שגיאה לוגית בתכנית, לנסות "לבצע" את הנתונים כאילו היו הוראות חוקיות. למשל, שגיאה שיכולה לגרום תופעה כזו הסתעפות לא נכונה. התכנית כמובן לא תעבוד נכון, אך לרוב הנזק הוא יותר חמור, כי נוצרת חריגת חומרה ברגע שהמעבד מבצע פעולה שאינה חוקית.

האסמבלר שלנו חייב להפריד, בקוד המכונה שהוא מייצר, בין קטע הנתונים לקטע ההוראות. **כלומר בקובץ הפלט (בקוד המכונה) תהיה הפרדה של הוראות ונתונים לשני קטעים נפרדים, ואילו בקובץ הקלט אין חובה שתהיה הפרדה כזו.** בהמשך מתואר אלגוריתם של האסמבלר, ובו פרטים כיצד לבצע את ההפרדה.

גילוי שגיאות בתכנית המקור

האסמבלר אמור לגלות ולדווח על שגיאות בתחביר של תכנית המקור, כגון פעולה שאינה קיימת, מספר אופרנדים שגוי, סוג אופרנד שאינו מתאים לפעולה, שם אוגר לא קיים, ועוד שגיאות אחרות. כמו כן מוודא האסמבלר שכל סמל מוגדר פעם אחת בדיוק.

מכאן, שכל שגיאה המתגלה על ידי האסמבלר נגרמת (בדרך כלל) על ידי שורת קלט מסוימת.

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

על כל הודעת שגיאה לציין גם את מספר השורה בתכנית המקור בה זוהתה השגיאה.

לתשומת לב: האסמבלר אינו עוצר את פעולתו אחרי שנמצאה השגיאה הראשונה, אלא ממשיך לעבור על הקלט כדי לגלות שגיאות נוספות, ככל שישנן. כמובן שאין כל צורך 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 הנתונה:

שם פעולה	שיטות מיעון חוקיות עבור אופרנד מקור	שיטות מיעון חוקיות עבור אופרנד יעד
mov	1,3,5	3,5
cmp	1,3,5	1,3,5
add	1,3,5	3,5
sub	1,3,5	3,5
not	אין אופרנד מקור	3,5
clr	אין אופרנד מקור	3,5
lea	3	3,5
inc	אין אופרנד מקור	3,5
dec	אין אופרנד מקור	3,5
jmp	אין אופרנד מקור	3,5
bne	אין אופרנד מקור	3,5
red	אין אופרנד מקור	3,5
prn	אין אופרנד מקור	1,3,5
jsr	אין אופרנד מקור	3,5
rts	אין אופרנד מקור	אין אופרנד יעד
stop	אין אופרנד מקור	אין אופרנד יעד

אלגוריתם שלדי של האסמבלר

לחידוד ההבנה של תהליך העבודה של האסמבלר, נציג להלן אלגוריתם שלדי למעבר הראשון ולמעבר השני. לתשומת לב: אין חובה להשתמש דווקא באלגוריתם זה.

אנו מחלקים את קוד המכונה לשני אזורים, אזור ההוראות (code) ואזור הנתונים (data).
לכל אזור יש מונה משלו, ונסמנם IC (מונה ההוראות - Instruction-Counter) ו-DC (מונה הנתונים - Data-Counter).

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

בכל מעבר מתחילים לקרוא את קובץ המקור מהתחלה.

מעבר ראשון

1. אתחל $DC \leftarrow 0$, $IC \leftarrow 0$.
2. קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עבור ל-16.
3. האם השדה הראשון הוא סמל? אם לא, עבור ל-5.
4. הדלק דגל "יש הגדרת סמל".
5. האם זוהי הנחיה לאחסון נתונים, כלומר, האם הנחית data או string? אם לא, עבור ל-8.
6. אם יש הגדרת סמל (תווית), הכנס אותו לטבלת הסמלים עם סימון (סמל מסוג data). ערכו יהיה DC (אם הסמל כבר נמצא בטבלה יש להודיע על שגיאה).
7. זהה את סוג הנתונים, קודד אותם בזיכרון, עדכן את מונה הנתונים DC בהתאם לאורכם, חזור ל-2.
8. האם זו הנחית extern או הנחית entry? אם לא, עבור ל-11.
9. האם זוהי הנחית extern? אם כן, הכנס כל סמל (אחד או יותר) המופיע כאופרנד של ההנחיה לתוך טבלת הסמלים ללא ערך, עם סימון (סמל מסוג external).
10. חזור ל-2.
11. אם יש הגדרת סמל, הכנס אותו לטבלת הסמלים עם סימון (סמל מסוג code). ערכו יהיה IC (אם הסמל כבר נמצא בטבלה יש להודיע על שגיאה).
12. חפש את שם הפעולה בטבלת שמות הפעולות, ואם לא נמצא – הודע על שגיאה בשם ההוראה.
13. נתח את מבנה האופרנדים של ההוראה וחשב את L. בנה כעת את הקוד הבינארי של המילה הראשונה של הפקודה.
14. עדכן $IC \leftarrow L + IC$.
15. חזור ל-2.
16. אם נמצאו שגיאות בקובץ המקור, עצור.
17. עדכן בטבלת הסמלים את ערכם של הסמלים מסוג data, ע"י הוספת הערך הסופי של IC (ראו הסבר בהמשך).
18. התחל מעבר שני.

מעבר שני

1. אתחל $IC \leftarrow 0$.
2. קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עבור ל-10.
3. אם השדה הראשון הוא סמל, דלג עליו.
4. האם זוהי הנחיה extern, string, data? אם כן, חזור ל-2.
5. האם זוהי הנחיה entry? אם לא, עבור ל-7.
6. סמן בטבלת הסמלים את הסמלים המתאימים כ-entry. חזור ל-2.
7. השלם את קידוד האופרנדים החל מהמילה השנייה בקוד הבינארי של ההוראה, בהתאם לשיטת המיעון. אם אופרנד הוא סמל, מצא את המען בטבלת הסמלים.
8. עדכן $IC \leftarrow IC + L$.
9. חזור ל-2.
10. אם נמצאו שגיאות במעבר שני, עצור.
11. צור ושמור את קבצי הפלט: קובץ קוד המכונה, קובץ סמלים חיצוניים, וקובץ סמלים של נקודות כניסה (ראו פרטים נוספים בהמשך).

נפעיל אלגוריתם זה על תוכנית הדוגמא שהצגנו קודם :

```

MAIN:      mov    @r3 ,LENGTH
LOOP:      jmp    L1
           prn    -5
           bne    LOOP
           sub    @r1, @r4
           bne    END
L1:         inc    K
           bne    LOOP
END:        stop
STR:        .string "abcdef"
LENGTH:     .data  6,-9,15
K:          .data  22

```

נבצע עתה מעבר ראשון על הקוד הנתון. נבנה את טבלת הסמלים. כמו כן, נבצע במעבר זה גם את קידוד כל הנתונים, וקידוד המילה הראשונה של כל הוראה. את החלקים שעדיין לא מתורגמים במעבר זה, נשאיר כמות שהם.

אנו מניחים שהקוד ייטען לזיכרון החל מהמען 100 (בבסיס דצימאלי).

Decimal Address	Source Code	Explanation	Binary Machine Code
0100 0101 0102	MAIN: mov @r3 ,LENGTH	first word of instruction source register 3 address of label LENGTH	101000001100 000110000000 ?
0103 0104	LOOP: jmp L1	address of label L1	000100101100 ?
0105 0106	prn -5	immediate value -5	000110000100 11111101100
0107 0108	bne LOOP	address of label LOOP	000101001100 ?
0109 0110	sub @r1, @r4	registers r1 and r4	101001110100 000010010000
0111 0112	bne END	address of label END	000101001100 ?
0113 0114	L1: inc K	address of label K	000011101100 ?
0115 0116	bne LOOP	address of label LOOP	000101001100 ?
0117	END: stop		000111100000
0118 0119 0120 0121 0122 0123 0124	STR: .string "abcdef"	ascii code 'a' ascii code 'b' ascii code 'c' ascii code 'd' ascii code 'e' ascii code 'f' ascii code '\0' (end of string)	000001100001 000001100010 000001100011 000001100100 000001100101 000001100110 000000000000
0125 0126 0127	LENGTH: .data 6,-9,15	integer 6 integer -9 integer 15	000000000110 111111110111 000000001111
0128	K: .data 22	integer 22	000000010110

טבלת הסמלים :

ערך (בבסיס דצימלי)	סמל
100	MAIN
103	LOOP
113	L1
117	END
118	STR
125	LENGTH
128	K

נבצע עתה את המעבר השני ונרשום את הקוד בצורתו הסופית :

Decimal Address	Source Code	Binary Machine Code
0100 0101 0102	MAIN: mov @r3 ,LENGTH	101000001100 000110000000 000111110110
0103 0104	LOOP: jmp L1	000100101100 000111000110
0105 0106	prn -5	000110000100 111111101100
0107 0108	bne LOOP	000101001100 000110011110
0109 0110	sub @r1, @r4	101001110100 000010010000
0111 0112	bne END	000101001100 000111010110
0113 0114	L1: inc K	000011101100 001000000010
0115 0116	bne LOOP	000101001100 000110011110
0117	END: stop	000111100000
0118 0119 0120 0121 0122 0123 0124	STR: .string "abcdef"	000001100001 000001100010 000001100011 000001100100 000001100101 000001100110 000000000000
0125 0126 0127	LENGTH: .data 6,-9,15	000000000110 111111110111 000000001111
0128	K: .data 22	000000010110

לאחר סיום עבודת האסמבלר, קבצי הפלט מועברים לשלבי הקישור והטעינה. לא נדון כאן באופן עבודת שלבי הקישור/טעינה (כאמור, אלה אינם למימוש בפרויקט זה). לאחר סיום שלבים אלה, התוכנית תהיה טעונה בזיכרון ומוכנה לריצה.

קבצי קלט ופלט של האסמבלר

בהפעלה של האסמבלר, יש להעביר אליו באמצעות ארגומנטים של שורת הפקודה (command-line arguments) רשימה של שמות קבצי מקור (אחד או יותר). אלו הם קבצי טקסט, ובהם תכניות בתחביר של שפת האסמבלי, שהוגדרה למעלה. האסמבלר פועל על כל קובץ

מקור בנפרד, ויוצר עבורו קובץ מטרה (object) נפרד המכיל את קוד המכונה. כמו כן האסמבלר יוצר קובץ externals עבור כל קובץ מקור בו יש הצהרות על סמלים חיצוניים, וכן קובץ entries עבור כל קובץ מקור בו יש הצהרות על סמלים כנקודות כניסה.

שמות קבצי המקור חייבים להיות בעלי הסיומת ".as". למשל, השמות x.as, y.as, ו-hello.as הם שמות חוקיים. העברת שמות הקבצים הללו כארגומנטים לאסמבלר נעשית ללא ציון הסיומת.

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

assembler x y hello

תגרום לכך שהאסמבלר יפעל על הקבצים: x.as, y.as, hello.as.

האסמבלר יוצר שמות לקבצי הפלט משם קובץ הקלט, כפי שהופיע בשורת הפקודה, בתוספת הסיומת ".ob". עבור קובץ ה-object, הסיומת ".ent". עבור קובץ ה-entries, והסיומת ".ext". עבור קובץ ה-externals.

מבנה כל קובץ פלט יתואר בהמשך.

לדוגמא: הפקודה: assembler x

תיצור את הקובץ x.ob וכן את הקבצים x.ent ו-x.ext אם קיימות הצהרות של entries/externals בקובץ המקור.

אופן פעולת האסמבלר

נרחיב כאן על אופן פעולת האסמבלר, בנוסף לאלגוריתם השלדי שניתן לעיל.

האסמבלר מחזיק שני מערכים, שייקראו להלן מערך ההוראות ומערך הנתונים. מערכים אלו נותנים למעשה תמונה של זיכרון המכונה (גודל כל כניסה במערך זהה לגודלה של מילת מכונה: 12 סיביות). במערך ההוראות מכניס האסמבלר את הקידוד של הוראות המכונה שנקראו במהלך המעבר על קובץ המקור. במערך הנתונים מכניס האסמבלר את קידוד הנתונים שנקראו מקובץ המקור (שורות מסוג .data, .string).

לאסמבלר יש שני מונים: מונה ההוראות (IC) ומונה הנתונים (DC). מונים אלו מצביעים על המקום הבא הפנוי במערכים לעיל, בהתאמה. כשמתחיל האסמבלר לעבור על קובץ מקור, שני מונים אלו מאופסים.

בנוסף יש לאסמבלר טבלה, אשר בה נאספות כל התוויות בהן נתקל האסמבלר במהלך המעבר על הקובץ. לטבלה זו קוראים טבלת סמלים (symbol-table). לכל סמל (תווית) נשמרים שמו, ערכו וטיפוסו (external או relocatable).

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

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

2. שורת הוראה:

האסמבלר מוצא מהי הפעולה, ומהן שיטות המיעון של האופרנדים. מספר האופרנדים אותם הוא מחפש נקבע בהתאם להוראה שנמצאה. האסמבלר קובע לכל אופרנד את ערכו באופן הבא:

- אם זה אוגר – האופרנד הוא מספר האוגר.
- אם זו תווית (מיעון ישיר) – האופרנד הוא ערך התווית כפי שמופיע בטבלת הסמלים (ייתכן והסמל טרם נמצא בטבלת הסמלים).
- אם זה מספר (מיעון מיידי) – האופרנד הוא המספר עצמו.
- אם זו שיטת מיעון אחרת – ערכו של האופרנד נקבע לפי המפרט של שיטת המיעון (ראו תאור שיטות המיעון לעיל)

קביעת שיטת המיעון נעשית בהתאם לתחביר של האופרנד, כפי שהוסבר לעיל בהגדרת שיטות המיעון. למשל, מספר מציין מיעון מיידי, תווית מציינת מיעון ישיר, שם של אוגר עם @ לפניו מציין מיעון אוגר, וכד'.

לאחר שהאסמבלר ניתח את השורה והחליט לגבי הפעולה, שיטת מיעון אופרנד המקור (אם יש), ושיטת מיעון אופרנד היעד (אם יש), הוא פועל באופן הבא :

אם זוהי פעולה בעלת שני אופרנדים, אזי האסמבלר מכניס למערך ההוראות, במקום עליו מצביע מונה ההוראות IC, את קוד המילה הראשונה של ההוראה (בשיטת הייצוג של הוראות המכונה כפי שתואר קודם לכן). מילה זו מכילה את קוד הפעולה, ואת שיטות המיעון. בנוסף "משריין" האסמבלר מקום במערך עבור המילים הנוספות הנדרשות עבור הוראה זו, ומגדיל את מונה ההוראות בהתאם. אם אחד או שני האופרנדים הם בשיטת מיעון אוגר או מיידי, האסמבלר יקודד כעת גם את המילים הנוספות הרלוונטיות במערך ההוראות.

אם זוהי פעולה בעלת אופרנד אחד בלבד, כלומר אין אופרנד מקור, אזי הקידוד הינו דומה לעיל, פרט לסיביות של שיטת המיעון של אופרנד המקור במילה הראשונה, אשר יכילו תמיד 0, מכיוון שאין רלוונטיות לפעולה.

אם זוהי פעולה ללא אופרנדים (rts, stop) אזי תקודד רק המילה הראשונה (והיחידה). הסיביות של שיטות המיעון של שני האופרנדים יכילו 0.

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

3. שורת הנחיה :

כאשר האסמבלר נתקל בהנחיה, הוא פועל בהתאם לסוג ההנחיה, באופן הבא :

I. 'data'.

האסמבלר קורא את רשימת המספרים, המופיעה לאחר 'data', מכניס כל מספר אל מערך הנתונים, ומקדם 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. הטיפול של התווית הוא relocatable, וכמו כן מסומן שההגדרה ניתנה בחלק הנתונים.

בסוף המעבר הראשון, ערך התווית יעודכן בטבלת הסמלים על ידי הוספת ה-IC (כלומר הוספת האורך הכולל של קידוד כל ההוראות). הסיבה לכך היא שבתמונת קוד המכונה, הנתונים מופרדים מההוראות, וכל הנתונים יופיעו אחרי כל ההוראות (ראה תאור קבצי הפלט בהמשך).

II. 'string'.

הטיפול ב-'string' דומה ל-'data', אלא שקודי ה-ascii של התווים הם אלו המוכנסים אל מערך הנתונים (כל תו בכניסה נפרדת). לאחר מכן מוכנס הערך 0 (המציין סוף מחרוזת) אל מערך הנתונים. מונה הנתונים מקודם באורך המחרוזת + 1 (כי גם האפס בסוף המחרוזת תופס מקום).

הטיפול בתווית המופיעה בשורה, זהה לטיפול הנעשה בהנחיה 'data'.

III. 'entry'.

זוהי בקשה לאסמבלר להכניס את התווית המופיעה כאופרנד של 'entry' אל קובץ ה-entries. האסמבלר רושם את הבקשה ובסיום העבודה, התווית הנ"ל תירשם בקובץ ה-entries.

IV. 'extern'.

זוהי הצהרה על סמל (תווית) המוגדר בקובץ אחר, ואשר קטע האסמבלי בקובץ הנוכחי עושה בו שימוש. האסמבלר מכניס את הסמל אל טבלת הסמלים. ערכו הוא 0 (הערך האמיתי לא ידוע, וייקבע רק בשלב הקישור), וטיפולו 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 תווית ואם באופן עקיף על ידי הנחיית extern).

פורמט קובץ ה-object

האסמבלר בונה את תמונת זיכרון המכונה כך שקידוד ההוראה הראשונה מקובץ האסמבלי **יכנס למען 100 (בבסיס 10)** בזיכרון, קידוד ההוראה השניה יכנס למען העוקב אחרי ההוראה הראשונה (תלוי במספר המילים של ההוראה הראשונה), וכך הלאה עד להוראה האחרונה.

מיד לאחר קידוד ההוראה האחרונה, מכניסים לתמונת הזיכרון את קידוד הנתונים שנבנו על ידי ההנחיות 'string' ו'.data'. הנתונים יוכנסו בסדר בו הם מופיעים בקובץ המקור. אופרנד של הוראה שמתייחס לסמל שהוגדר באותו קובץ, יקודד כך שיצביע על המקום המתאים בתמונת הזיכרון שבונה האסמבלר.

נשים לב שהמשתנים מופיעים בתמונת הזיכרון אחרי ההוראות. זוהי הסיבה בגללה יש לעדכן בטבלת הסמלים, בסוף המעבר הראשון, את ערכי הסמלים המגדירים נתונים (סמלים מסוג data).

עקרונית, קובץ object מכיל את תמונת הזיכרון שתוארה כאן. קובץ object מורכב משורות של טקסט כדלקמן.

השורה הראשונה היא כותרת המכילה שני מספרים בבסיס 10: האורך הכולל של קטע ההוראות (במילות זיכרון), ואחרי האורך הכולל של קטע הנתונים (במילות זיכרון). בין שני המספרים יש רווח אחד

השורות הבאות מכילות את תוכן הזיכרון. בכל שורה מופע תוכן של מילה אחת, לפי הסדר החל מהמילה בכתובת 100. תוכן המילה מקודד בשיטת Base64.

ניתן לקרוא על שיטת Base64 בקישור הבא: <https://en.wikipedia.org/wiki/Base64>

מילה היא בגודל 12 ביטים. כל 6 ביטים יומרו לתו מתאים לפי הטבלה המגדירה את Base64. לפיכך בכל שורה בקובץ object (מלבד השורה הראשונה) יהיו בדיוק שני תווים בקוד Base64.

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

פורמט קובץ ה-entries

קובץ ה-entries בנוי משורות טקסט. כל שורה מכילה שם של סמל שהוגדר כ- entry ואת ערכו, כפי שנמצא בטבלת הסמלים. הערכים מיוצגים בבסיס 10.

פורמט קובץ ה-externals

קובץ ה-externals בנוי אף הוא משורות טקסט. כל שורה מכילה שם של סמל שהוגדר external, וכתובת בקוד המכונה בה יש קידוד של אופרנד המתייחס לסמל זה. כמוכן שייתכן ויש מספר כתובות בקוד המכונה בהם מתייחסים לאותו סמל חיצוני. לכל התייחסות כזו תהיה שורה נפרדת בקובץ ה-externals. הכתובות מיוצגות בבסיס 10

להלן קובץ מקור (קלט) לדוגמה, בשם ps.as

```
; file ps.as

.entry LENGTH
.extern W
MAIN:      mov    @r3,LENGTH
LOOP:      jmp    L1
           prn    -5
           bne    W
           sub    @r1,@r4
           bne    L3
L1:         inc    K
.entry LOOP
           jmp    W
END:        stop
STR:        .string "abcdef"
LENGTH:     .data  6,-9,15
K:          .data  22
.extern L3
```

להלן טבלת הקידוד המלא הבינארי שמתקבל מהפעלת האסמבלר על קובץ המקור, ולאחריה הפורמטים של קבצי הפלט השונים הנבנים על ידי האסמבלר.

Decimal Address	Source Code	Binary Machine Code
0100 0101 0102	MAIN: mov @r3,LENGTH	101000001100 000110000000 000111110110
0103 0104	LOOP: jmp L1	000100101100 000111000110
0105 0106	prn -5	000110000100 111111101100
0107 0108	bne W	000101001100 000000000001
0109 0110	sub @r1,@r4	101001110100 000010010000
0111 0112	bne L3	000101001100 000000000001
0113 0114	L1: inc K	000011101100 001000000010
0115 0116	jmp W	000100101100 000000000001
0117	END: stop	000111100000
0118 0119 0120 0121 0122 0123 0124	STR: .string "abcdef"	000001100001 000001100010 000001100011 000001100100 000001100101 000001100110 000000000000
0125 0126 0127	LENGTH: .data 6,-9,15	000000000110 111111110111 000000001111
0128	K: .data 22	000000010110

הקובץ ps.ob:

נבחין כי בדוגמה לעיל, קטע ההוראות בגודל 18 מילים, ואילו קטע הנתונים בגודל 11 מילים. להלן קובץ הפלט המתאים לדוגמה זו, מקודד בשיטת Base64 (סה"כ 30 שורות, כולל הכותרת):

```
18 11
oM
GA
H2
ES
HG
GE
/s
FM
AB
p0
CQ
FM
AB
DS
IC
ES
AB
Hg
Bh
Bi
Bj
Bk
Bl
Bm
AA
AG
/3
AP
AW
```

הקובץ ps.ent:

כל ערכי הסמלים מיוצגים בבסיס 10.

```
LOOP      103
LENGTH    125
```

הקובץ ps.ext:

כל הכתובות מיוצגות בבסיס 10.

```
W      108
L3     112
W      116
```

לתשומת לב: אם בקובץ המקור אין הצהרת extern, אזי לא ייווצר עבורו קובץ ext. בדומה, אם אין בקובץ המקור הצהרת entry, לא ייווצר קובץ ext. אין ליצור קובץ ext או ent שנשאר ריק.

הערה: אין חשיבות לסדר השורות בקבצים מסוג ent או ext. כל שורה עומדת בפני עצמה.

סיכום והנחיות כלליות

- אורך התוכנית, הניתנת כקלט לאסמבלר, אינו ידוע מראש (ואינו קשור לגודל 1024 של הזיכרון במעבד הדמיוני). ולכן אורכה של התוכנית המתורגמת אינו אמור להיות צפוי מראש. אולם בכדי להקל במימוש התכנית, ניתן להניח גודל מקסימלי. לפיכך יש אפשרות להשתמש במערכים לאחסון קוד המכונה בלבד. מבני נתונים אחרים בפרויקט (למשל טבלת הסמלים), יש לממשל על פי יעילות/תכונות נדרשות (למשל באמצעות הקצאת זיכרון דינמי ורשימה מקושרת).
- קבצי הפלט של האסמבלר צריכים להיות בפורמטים המופיעים למעלה. שמם של קבצי הפלט צריך להיות תואם לשם קובץ הקלט, פרט לסיומות. למשל, אם קובץ הקלט הוא prog.as אזי קבצי הפלט שיווצרו הם : prog.ob, prog.ext, prog.ent .
- אופן הפעלת האסמבלר צריך להיות תואם לנדרש בממ"ן, ללא שינויים כלשהם. אין להוסיף תפריטים אינטראקטיביים למיניהם, וכד'. הפעלת התוכנית תיעשה רק עם ארגומנטים של שורת פקודה.
- יש להקפיד לחלק את מימוש האסמבלר למספר מודולים (קבצים בשפת C) לפי משימות. אין לרכז משימות מסוגים שונים במודול יחיד. מומלץ לחלק למודולים כגון : מבני נתונים, מעבר ראשון, מעבר שני, פונקציות עזר (למשל, תרגום לבסיס, ניתוח תחבירי של שורה), טבלת סמלים, ועוד.
- יש להקפיד ולתעד את המימוש באופן מלא וברור, באמצעות הערות מפורטות בקוד.
- יש להקפיד להתעלם מרווחים וטאבים, ולהיות סלחנים כלפי תוכנית קלט העושה שימוש ביותר רווחים מהנדרש. למשל, אם בהוראה יש שני אופרנדים המופרדים בפסיק, אזי לפני או אחרי הפסיק יכולים להיות רווחים וטאבים בכל כמות. בדומה, גם לפני ואחרי שם ההוראה. בכל המקרים האלה השורה צריכה להיחשב חוקית (לפחות מבחינת הרווחים).
- במקרה שתוכנית הקלט בשפת אסמבלי מכילה שגיאות תחביריות, נדרש לגלות ולדווח, כמו באסמבלר אמיתי, על כל השגיאות הקיימות, ולא לעצור לאחר גילוי השגיאה הראשונה. כמובן שאם הקלט מכיל שגיאות, אין טעם להפיק קבצי פלט (ob, ext, ent).

תם ונשלם פרק ההסברים והגדרת הפרויקט.

בשאלות ניתן לפנות לקבוצת הדיון באתר הקורס, ואל כל אחד מהמנחים בשעות הקבלה שלהם.

להזכירכם, באפשרותו של כל סטודנט לפנות לכל מנחה, לאו דווקא למנחה 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 לכולם.

לתשומת לבכם : לא תינתן דחיה בהגשת הממ"ן, פרט למקרים מיוחדים (כגון מילואים או מחלה). במקרים כאלה יש לבקש ולקבל אישור מראש מצוות הקורס.

ב ה צ ל ח ה !