

האוניברסיטה הפתוחה

20465

מעבדה בתכנות מערכות

חוברת הקורס – סתיו 2017א

כתבה: מיכל אבימור

אוקטובר 2016 – סמסטר סתיו – תשע"ז

פנימי – לא להפצה.

© כל הזכויות שמורות לאוניברסיטה הפתוחה.

תוכן העניינים

א	אל הסטודנט
ב	1. לוח זמנים ופעילויות
ד	2. תיאור המטלות
ו	3. התנאים לקבלת נקודות זכות
1	ממ"ן 11
3	ממ"ן 12
5	ממ"ן 22
9	ממ"ן 23
11	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

הספרייה 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, והמממ"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 (20465 /א/2017)

שבוע הלימוד	תאריכי שבוע הלימוד	יחידת הלימוד המומלצת	מפגשי ההנחיה*	תאריך אחרון למשלוח הממ"ן (למנחה)
1	4.11.2016-30.10.2016	ספר C פרקים 1-2-3	מפגש ראשון	
2	11.11.2016-6.11.2016	ספר C פרקים 1-2-3		
3	18.11.2016-13.11.2016	ספר C פרק 4	מפגש שני	
4	25.11.2016-20.11.2016	ספר C פרק 4		
5	2.12.2016-27.11.2016	ספר C פרק 5	מפגש שלישי	ממ"ן 11 27.11.2016
6	9.12.2016-4.12.2016	ספר C פרק 5		
7	16.12.2016-11.12.2016	ספר C פרק 6	מפגש רביעי	
8	23.12.2016-18.12.2016	ספר C פרק 6		ממ"ן 12 18.12.2016

* 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	30.12.2016-25.12.2016 (א-ו חנוכה)	ספר C פרק 6		
10	6.1.2017-1.1.2017 (א חנוכה)	ספר C פרק 7	מפגש חמישי	
11	13.1.2017-8.1.2017	ספר C פרק 7		ממ"ן 22 8.1.2017
12	20.1.2017-15.1.2017	ספר C פרק 8 + פרויקט	מפגש שישי	
13	27.1.2017-22.1.2017	פרויקט וחזרה		
14	3.2.2017-29.1.2017	פרויקט וחזרה	מפגש שביעי	ממ"ן 23 29.1.2017
15	10.2.2017-5.2.2017	פרויקט וחזרה		ממ"ן 14** 2.4.2017

מועדי בחינות הגמר יפורסמו בנפרד

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

** לא תינתן דחייה בהגשת הפרויקט, פרט למקרים של מילואים או מחלה, 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 יכול:

- 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 בדיעון של האו"פ. רק את ממ"ן 14 מותר להגשה בזוגות (לא ניתן להגיש בשלוש!), 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

מספר השאלות: 2 משקל המטלה: 4 נקודות (חובה)

מועד אחרון להגשה: 27.11.2016 סמסטר: 2017א'

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` נדרש להגיש את קבצי המקור (`.c`, `.h`), קבצי ההרצה (את קבצי `.o` אין צורך לצרף), קבצי הסביבה המתאימים (כולל קבצי `MAKEFILE`), קבצי קלט וקבצי פלט (לפי הנחייה במפגש / באתר). נדרש ששם הספרייה ושם הקובץ לריצה יהיו כשם קובץ התוכנית הראשית, ללא הסיומת `.c`. יש להגיש תכניות מלאות (בין השאר מכילות `main`), הניתנות להידור והרצה ובדיקת הנדרש, ללא תוספות קוד. 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_scalar.c`) (50 נקודות)

יהיו u ו- v וקטורים (n ממדים):

$$v = (v_1, v_2, v_3, \dots, v_n)$$

ו-

$$u = (u_1, u_2, u_3, \dots, u_n)$$

המכפלה הפנימית (או המכפלה הסקלרית) של u ו- v , היא הסקלר המתקבל על ידי חיבור

מכפלות הרכיבים המתאימים של שני הוקטורים:

$$u \cdot v = u_1 \cdot v_1 + u_2 \cdot v_2 + u_3 \cdot v_3 + \dots + u_n \cdot v_n$$

דוגמה:

יהיו -

$$u = (1, -2, 3, -4)$$

$$v = (6, 7, 1, -2)$$

$$w = (5, -4, 5, 7)$$

אזי

$$u \cdot v = 1 \cdot 6 + (-2) \cdot 7 + 3 \cdot 1 + (-4) \cdot (-2) = 3$$

$$u \cdot w = 1 \cdot 5 + (-2) \cdot (-4) + 3 \cdot 5 + (-4) \cdot 7 = 0$$

עליכם לכתוב תכנית, המכילה פונקציה בשם `scalar_product` אשר מקבלת כפרמטרים שני וקטורים ומספר שלם אחד (מה תפקידו?), מחשבת את המכפלה הסקלרית של שני הוקטורים, ומחזירה אותו.

הערה: ניתן להניח שבכל קריאה לפונקציה שני הפרמטרים (הוקטורים) יהיו בעלי אורך זהה, אולם על הפונקציה להיות מסוגלת לטפל בזוג וקטורים באורך כלשהו.

שאלה 2 (תכנית ראשית בקובץ `con.c`) (50 נקודות)

עליכם לכתוב תכנית, המכילה פונקציה `void contract (char s1[], char s2[])` המקבלת ב-`s1` סדרת תווים, ומחזירה ב-`s2` את אותה סדרה בפורמט מקוצר כלהלן: `abcdef` יתורגם ל: `a-f`. וכך כל סדרה רצופה של תווי `ascii`.

הערות:

א. ניתן להניח שאורך המערך `s2` גדול או שווה לאורך המערך `s1`.

ב. במקרה של רצף תווים באורך 2 (למשל `xy`) יש להשאיר את המקור (בדוגמא, אין צורך לתרגם ל- `x-y`).

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

משקל המטלה: 5 נקודות (חובה)

מספר השאלות: 1

מועד אחרון להגשה: 18.12.2016

סמסטר: 2017א'

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** - נדרש להגיש את קבצי המקור (.c, .h), קבצי ההרצה (את קבצי .o אין צורך לצרף), קבצי הסביבה המתאימים (כולל קבצי MAKEFILE), קבצי קלט וקבצי פלט (לפי הנחייה במפגש / באתר). נדרש ששם הספרייה ושם הקובץ לריצה יהיו כשם קובץ התוכנית הראשית, ללא הסיומת .c. יש להגיש תכניות מלאות (בין השאר מכילות main), ניתנות להידור/הרצה/בדיקת הנדרש, ללא תוספות קוד.

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 summary.c)

עליכם לכתוב פונקציה, המקבלת, כפרמטר, מערך של מספרים שלמים, ואת גודל המערך. על הפונקציה להחזיר "סיכום" המערך, כאשר כל מספר, במערך המקורי, מוחלף בסכום האיברים עד אליו. **אסור** לפונקציה לגרום לשינויים, על המערך המקורי.

לדוגמה:

בהינתן המערך הבא:

3 5 1 10 14

הפונקציה תחזיר את המערך הבא:

3 8 9 19 33

עליכם לכתוב תכנית מחשב אינטראקטיבית, הקוראת את המערך ואת גודלו מהקלט הסטנדרטי, ומדפיסה לפלט הסטנדרטי את ה"סיכום" שלו.

הערה: בפלט ובקלט יש להשתמש בייצוג עשרוני.

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: 2017 מועד אחרון להגשה: 8.1.2017

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** נדרש להגיש את קבצי המקור (.c, .h), קבצי ההרצה (את קבצי .o אין צורך לצרף), קבצי הסביבה המתאימים (כולל קבצי MAKEFILE), קבצי קלט וקבצי פלט (לפי הנחייה במפגש / באתר). נדרש ששם הספרייה ושם הקובץ לריצה יהיו כשם קובץ התוכנית הראשית, ללא הסיומת .c. יש להגיש תכניות מלאות (בין השאר מכילות main), ניתנות להידור/הרצה/בדיקת הנדרש, ללא תוספות קוד. 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 (קבצים ראשיים set.c, set.h, main.c)

עליכם לכתוב תכנית מחשב אינטראקטיבית הקוראת פקודות, מפענחת ומבצעת אותן. הפקודות עוסקות בפעולות על קבוצות. תזכורת: קבוצה - set - היא אוסף של איברים, ברי מניה, שכל איבר בו מופיע לכל היותר פעם אחת.

עליכם להגדיר, תוך השימוש בפקודת typedef את הטיפוס set, אשר מסוגל להחזיק קבוצת שלמים בתחום [0...127]. על מבנה הנתונים שבחרת להיות יעיל מבחינת כמות זיכרון, הנדרשת לשמירתו (רמז: עבודה עם סיביות). בנוסף עליכם להגדיר 6 משתנים חיצוניים A,B,C,D,E,F מטיפוס זה. כל שם קבוצה בפקודות שלהלן יילקח מתוך השישה הנ"ל.

הפקודות המותרות כקלט לתכנית:

1. רשימת-ערכים-מופרדים-על-ידי-פסיקים, שם-קבוצה read_set

הפקודה תגרום לקריאת הערכים ברשימה לתוך הקבוצה ששמה ניתן בפקודה. רשימת הערכים אינה חייבת להיות סדורה ומותר לערך להופיע בה יותר מפעם אחת.
לדוגמה: הפקודה הבאה:

```
read_set A,5, 6, 5, 76, 44, 23, 23, 98, 23, -1
```

מייצגת את הקבוצה הבאה:

5, 6, 23, 44, 76, 98

סוף הקבוצה מצוין ע"י המספר 1-.

2. שם-קבוצה print_set

הקבוצה ששמה ניתן תודפס בצורה נאה וסדורה, לכל היותר 16 אברים בשורת פלט אחת. במקרה של הקבוצה הריקה, יש לכתוב זאת במילים.

3. שם-קבוצה-ג', שם-קבוצה-ב', שם-קבוצה-א' union_set

מבצעת איחוד של קבוצה א' וקבוצה ב' ואת התוצאה מאחסנת בקבוצה ג'.

4. שם-קבוצה-ג', שם-קבוצה-ב', שם-קבוצה-א' intersect_set

מבצעת חיתוך של קבוצה א' וקבוצה ב' ואת התוצאה מאחסנת בקבוצה ג'.

5. שם-קבוצה-ג', שם-קבוצה-ב', שם-קבוצה-א' sub_set

מבצעת חיסור של קבוצה ב' מקבוצה א' ואת התוצאה מאחסנת בקבוצה ג'.
תזכורת: חיסור של קבוצה ב' מקבוצה א' זה כל האיברים הנמצאים בקבוצה א' ולא נמצאים בקבוצה ב'.

6. halt:

התכנית תפסיק לרוץ ותצא לרמת מערכת ההפעלה.
התכנית צריכה לתת סימן (prompt) על המסך שמודיע שהיא מוכנה לקבל קלט.
התכנית תמשיך לעבוד עד שתקבל את פקודת halt.
התכנית אינה מניחה נכונות הקלט ויש להודיע על שגיאות.

דוגמאות:

לפקודה:

```
read_set W, 3, 6, 5, 4, 4, -1
```

יש להגיב בהודעה:

```
"No such set"
```

לפקודה:

```
mba A,B,C
```

יש להגיב בהודעה:

```
"No such command"
```

לפקודה:

```
read_set A, 45,567,34
```

יש להגיב:

```
"567 is out of range"
```

וכו'...

קלט לדוגמה:

```
read_set A,45,23,6,7,4,3,75,34,-1
```

```
read_set B,5,4,3,2,78,45,43,-1
```

```
union_set A, B, C
```

```
intersect_set A, B, C
```

```
sub_set A, B, D
```

```
print_set A
```

```
print_set B
```

```
print_set C
```

```
print_set D
```

```
print_set E
```

```
halt
```

יש לחלק את התוכנית למספר קבצים: set.h, set.c, main.c.

בקובץ set.c יש לרכז את הפונקציות החישוביות ובקובץ main.c את פעילויות האינטראקציה.

קובץ set.c מייצא את אבות הטיפוס של הפונקציות הקיימות בו באמצעות set.h.

יש לכתוב לתכנית ממשק פשוט, כך שהמשתמשים יוכלו להבין מה לעשות, על מנת להריץ את התוכנית.

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: 2017'א מועד אחרון להגשה: 29.1.2017

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** - נדרש להגיש את קבצי המקור (.c, .h), קבצי ההרצה (את קבצי .o אין צורך לצרף), קבצי הסביבה המתאימים (כולל קבצי MAKEFILE), קבצי קלט וקבצי פלט (לפי הנחייה במפגש / באתר). נדרש ששם הספרייה ושם הקובץ לריצה יהיו כשם קובץ התוכנית הראשית, ללא הסיומת .c. יש להגיש תכניות מלאות (בין השאר מכילות main), ניתנות להידור/הרצה/בדיקת הנדרש, ללא תוספות קוד. 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)

בכל סעיף, עליכם לכתוב האם "תמיד נכון", בשפת C, "לפעמים נכון ולפעמים אינו נכון" או "תמיד אינו נכון". עליכם לנמק את תשובתכם, תשובה לא מנומקת, גם אם היא נכונה, לא תזכה בנקודות.

- 5 נק') א. בשפת ANSI-C אין לנו שליטה על אופן הקצאת המערך בזיכרון, ולא נוכל להסתמך על כך שהמערך יופיע תמיד כגוש זיכרון רציף.
- 5 נק') ב. מכיוון ש-argv הוא בעצם מצביע למערך, ניתן לבצע עליו פעולות אריתמטיות של מצביעים.

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

עליכם לכתוב תכנית המכילה פונקציה, המקבלת מספר משתנה של ארגומנטים, כולם מספרים שלמים בתחום 0 עד 100 + , כל רשימת ארגומנטים תסתיים ב-1. הפונקציה תחזיר בכל קריאה אליה, את המספר הקטן ביותר שנשלח עד כה בכל הקריאות אליה.

למשל, אם בקריאה הראשונה נשלחו המספרים : 1 - 20 5 78 90 אזי יוחזר 5.

אם בקריאה הבאה לפונקציה נשלחו המספרים : 1 - 2 40 70 אזי יוחזר 2

ואם בקריאה השלישית נשלחו המספרים : 1 - 30 40 אזי יוחזר 2 עדיין.

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 : 2017' מועד אחרון להגשה : 2.4.2017

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.

עליכם לכתוב תוכנת אסמבלר, לשפת אסמבלי שתוגדר בהמשך. הפרויקט ייכתב בשפת C. אין להוסיף ספריות חיצוניות: ניתן להשתמש רק בספריות, מתוך הספרייה הסטנדרטית.

עליכם להגיש :

1. קבצי המקור של התוכנית שכתבת (קבצים בעלי סיומת c או h).
2. קבצי הרצה.
3. הגדרת סביבת העבודה (MAKEFILE). יש לקמפל עם הדגלים : -Wall -ansi -pedantic ולנפות את כל ההערות שמוציא הקומפיילר, כך שהתכנית תתקמפל ללא הערות.
4. דוגמאות של קבצי קלט, וקבצי הפלט שנוצרו על ידי הפעלת האסמבלר על קבצי קלט אלה.
5. דוגמאות של הפעלת האסמבלר על קבצי קלט המכילים מגוון של שגיאות אסמבלי (ולכן לא נוצרים קבצי פלט). יש לצרף את תדפיסי המסך המראים את הודעות השגיאה שמוציא האסמבלר.

בשל גודל הפרויקט, עליכם לחלק את התוכנית למספר קבצי מקור. יש להקפיד שהקוד הנמצא בתוכניות המקור יעמוד בקריטריונים של בהירות, קריאות וכתובה נכונה.

נזכיר מספר היבטים חשובים :

1. הפשטה של מבני הנתונים : רצוי (במידת האפשר) להפריד בין הגישה למבני הנתונים לבין המימוש של מבנה הנתונים. כך, למשל, בעת כתיבת שגרות לטיפול במחסנית, אין זה מעניינם של המשתמשים בשגרות אלה, אם המחסנית ממומשת באמצעות מערך או באמצעות רשימה מקושרת.
2. קריאות הקוד : רצוי להצהיר על הקבועים הרלוונטיים בנפרד, תוך שימוש בפקודת `#define`, ולהימנע מ"מספרי קסם", שמשמעותם נהירה לך בלבד.
3. תיעוד : יש להכניס בקבצי המקור תיעוד תמציתי וברור, שיסביר תפקידה של כל פונקציה ופונקציה. כמו כן יש להסביר את תפקידם של משתנים חשובים.

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.

הפרויקט כולל כתיבה של תוכנית אסמבלר עבור שפת אסמבלי, שהוגדרה במיוחד עבור פרויקט זה. מותר לעבוד בזוגות. אין לעבוד בצוות גדול יותר משניים. **פרויקטים שיוגשו בשלישיות או יותר לא יבדקו.** חובה ששני סטודנטים, הבוחרים להגיש יחד את הפרויקט, יהיו **שייכים לאותה קבוצה.**

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, הנקראות בתים. כאשר נמצאת בזיכרון תוכנית משתמש, לא ניתן להבחין, בעין שאינה מיומנת, בהבדל פיסי כלשהו, בין אותו חלק בזיכרון, שבו נמצאת התכנית, לבין שאר הזיכרון.

יחידת העיבוד המרכזית (היע"מ) יכולה לבצע מספר מסוים של הוראות פשוטות, ולשם כך היא משתמשת בזיכרון המחשב ובכמה אוגרים (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), תועבר בסופו של דבר באמצעות תוכנה מיוחדת לצורה סופית זו.

קוד בשפת מכונה הוא רצף של ביטים המהווים קידוד של סדרת הוראות (תכנית) 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 שהתכנית 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 לבך: בהסברים הכלליים על אופן עבודת תוכנת האסמבלר, תהיה מדי פעם התייחסות גם לעבודת שלבי הקישור והטעינה. התייחסויות אילו נועדו על מנת לאפשר לך להבין את המשך תהליך העיבוד של הפלט של תוכנת האסמבלר. אין לטעות: עליכם לכתוב את תוכנת האסמבלר בלבד, **אין** צורך לכתוב גם את תוכנת 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 מלת זיכרון במחשב הוא 15 סיביות. האריתמטיקה נעשית בשיטת המשלים ל-2 (2's complement). מחשב זה מטפל רק במספרים שלמים חיוביים ושליילים, אין טיפול במספרים ממשיים.

אוגרים:

למעבד 8 אוגרים כלליים (r0, r1, r2, r3, r4, r5, r6, r7). גודלו של כל אוגר הוא 15 סיביות. הסיבית הכי פחות משמעותית תצויין כסיבית מס' 0, והסיבית המשמעותית ביותר כמס' 14.

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

גודל הזיכרון הוא 1000 תאים, בכתובות 0-999 (בבסיס עשרוני), וכל תא הוא בגודל של 15 סיביות. לתא בזיכרון נקרא גם בשם "מילה". הסיביות בכל מילה ממוספרות בדומה לאוגר, כמפורט לעיל.

קידוד של תווים (characters) נעשה בקוד ascii.

אפיון מבנה הוראת מכונה:

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

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
לא בשימוש (תמיד יהיו 111)					group			opcode	מיעון אופרנד מקור		מיעון אופרנד יעד		E,R,A	

קידוד ההוראות ייעשה לבסיס 16 (הקסאדצימלי), ע"י השלמת אפסים במקרה הצורך.

סיביות 6-9 מהוות את קוד ההוראה (opcode). בשפה שלנו יש 16 קודי הוראה והם:

הקוד בבסיס דצימלי (10)	פעולה
0	mov
1	cmp
2	add
3	sub
4	not
5	clr
6	lea
7	inc

8	dec
9	jmp
10	bne
11	red
12	prn
13	jsr
14	rts
15	stop

ההוראות נכתבות תמיד באותיות קטנות. פרוט משמעות ההוראות יבוא בהמשך.

סיביות 0-1 (A,R,E)

סיביות אלה מראות 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, והן מתווספות גם למילים הנוספות שיש לקידודים אלה.

סיביות 2-3 מקודדות את מספרה של שיטת המיעון של אופרנד היעד (destination operand).

סיביות 4-5 מקודדות את מספרה של שיטת המיעון של אופרנד המקור (source operand).

בשפה שלנו קיימות ארבע שיטות מיעון, שמספרן הוא בין 0 ל-3.

השימוש בשיטות מיעון מצריך קידוד של מילות-מידע נוספות. אם שיטת המיעון של רק אחד משני האופרנדים, דורשת מילות מידע נוספות, אזי מילות המידע הנוספות מתייחסות לאופרנד זה. אך אם שיטת המיעון של שני האופרנדים דורשות מילות-מידע נוספות, אזי מילות-המידע הנוספות הראשונות מתייחסות לאופרנד המקור (source operand) ומילות-המידע הנוספות האחרונות מתייחסות לאופרנד היעד (destination operand).

גם למילות-המידע הנוספות יהיו זוג סיביות בצד ימין, המציינות את השדה A,R,E

סיביות 6-9 מהוות את קוד ההוראה כפי שהוסבר קודם.

סיביות 10-11 (group)

סיביות אלו מסמלות את סוג ההוראה המקודדת. בשפה קיימות הוראות ללא אופרנדים, הוראות עם אופרנד יחיד, והוראות עם 2 אופרנדים.

ערכי סיביות אלה יהיו 00 עבור קידוד הוראה ללא אופרנדים. 01 עבור קידוד הוראה עם אופרנד יחיד, ו-10 עבור קידוד הוראה עם 2 אופרנדים.

סיביות 12-14 – לא בשימוש וערכן תמיד יהיה 111

ארבע שיטות המיעון הקיימות במכונה שלנו הן :

ערך	שיטת מיעון	תוכן המילה נוספת	אופן הכתיבה	דוגמא
0	מיעון מידי	המילה הנוספת של הפקודה מכילה את האופרנד עצמו, שהוא מספר המיוצג ב- 13 סיביות אליהם מתווספות זוג סיביות של שדה A,R,E	האופרנד מתחיל בתו # ולאחריו ובצמוד אליו מופיע מספר שלם בבסיס עשרוני	mov #-1,r2 בדוגמה זו האופרנד הראשון של הפקודה נתון בשיטת מיעון מידי. הפקודה כותבת את הערך 1- לתוך אוגר r2
1	מיעון ישיר	המילה הנוספת של הפקודה מכילה מען של מילה בזיכרון. מילה זו בזיכרון הינה האופרנד. המען מיוצג ב- 13 סיביות אליהן מתווספות זוג סיביות של שדה A,R,E	האופרנד הינו <u>תווית</u> שהוצהרה או תוצהר בהמשך הקובץ. ההצהרה נעשית על ידי כתיבת תווית בקובץ (בפקודת '.data' או '.string' או בהגדרת תווית ליד שורת קוד של התוכנית). או על ידי הנחת 'extern'	dec x בדוגמה זו, תוכן המילה שבכתובת x בזיכרון (ה"משתנה" x) מוקטן ב-1.
2	מיעון אינדקס באמצעות אוגר	בשיטת מיעון זו משתתפים שני אוגרים. בזמן ביצוע ההוראה, המעבד מחבר את תוכן שני האוגרים ומשתמש בתוצאה כמען בשל מילה בזיכרון. מילה זו בזיכרון היא האופרנד. במילה הנוספת של הפקודה, ששת הסיביות 2-7 מקודדים את מספר האוגר הרשום לפני הסוגריים המרובעים, וששת הסיביות 8-13 מקודדים את האוגר הרשום בתוך הסוגריים. לייצוג זה מתווספות זוג סיביות של שדה A,R,E. סיבית 14 תכיל 0.	האופרנד מורכב משם של אוגר אי-זוגי לפני הסוגריים המרובעים, ושם של אוגר זוגי בתוך הסוגריים המרובעים.	clr r5[r2] בדוגמה זו, הפקודה מאפסת את תוכן המילה בזיכרון שכתובתה נתונה על ידי סכום האוגרים r5 ו-r2. למשל, אם אוגר r5 מכיל את המספר 500 (עשרוני), והאוגר r2 מכיל את המספר 8, הכתובת בזיכרון היא 508.
3	מיעון אוגר ישיר	האופרנד הוא אוגר. אם האוגר משמש כאופרנד יעד, במילה נוספת של הפקודה תכיל בששת הסיביות 2-7 את מספרו של האוגר. אם האוגר משמש כאופרנד מקור, הוא יקודד במילה נוספת שתכיל בששת הסיביות 8-13 את מספרו של האוגר. אם בפקודה יש שני אופרנדים ושניהם אוגרים, הם יחלקו מילה נוספת משותפת. כאשר הסיביות 2-7 הן עבור אוגר היעד, והסיביות 8-13 הן עבור אוגר המקור. לייצוג זה מתווספות זוג סיביות של שדה A,R,E. שדה שאינו בשימוש יכיל 0. סיבית 14 תכיל תמיד 0.	האופרנד הינו שם של אוגר.	mov r1,r2 בדוגמה זו, אופרנד המקור הוא האוגר r1, ואופרנד היעד הוא האוגר r2. הפקודה מעתיקה את תוכן אוגר r1 לתוך אוגר r2.

הערה: מותר להתייחס לתווית עוד לפני שמצהירים עליה (באופן סתום או מפורש), בתנאי שהיא
אכן מוצהרת במקום כלשהו בקובץ.

אפיון הוראות המכונה :

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

קבוצה ההוראות הראשונה :

ההוראות הזקוקות לשני אופרנדים. הפקודות השייכות לקבוצה זו הן :

mov, cmp, add, sub, lea

הסבר דוגמא	דוגמא	הסבר פעולה	פקודה
העתק תוכן משתנה A לאוגר r1.	mov A, r1	מבצעת העתקה של האופרנד הראשון, אופרנד המקור (source) אל האופרנד השני, אופרנד היעד (destination) (בהתאם לשיטת המיעון).	mov
אם תוכן הערך הנמצא במען A זהה לתוכנו של אוגר r1 אזי דגל האפס, Z, באוגר הסטטוס (PSW) יודלק, אחרת הדגל יאופס.	cmp A, r1	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
אוגר r0 מקבל את סכום תוכן משתנה A וערכו הנוכחי של אוגר r0.	add A, r0	אופרנד היעד (השני) מקבל את סכום אופרנד המקור (הראשון) והיעד (השני).	add
אוגר r1 מקבל את תוצאת החיסור של הערך 3 מתוכן האוגר, r1.	sub #3, r1	אופרנד היעד (השני) מקבל את תוצאת החיסור של אופרנד המקור (הראשון) מאופרנד היעד (השני).	sub
המען של תווית HELLO מוכנס לאוגר r1.	lea HELLO, r1	lea הינו ראשי תיבות של load effective address. פעולה זו מבצעת טעינה של המען בזיכרון המצוין על ידי התווית שבאופרנד הראשון (המקור), אל אופרנד היעד, (האופרנד השני).	lea

קבוצת ההוראות השניה :

הוראות הדורשות אופרנד אחד בלבד. במקרה זה זוג הסיביות (4-5) חסרות משמעות מכיוון שאין אופרנד מקור (אופרנד ראשון) אלא רק אופרנד יעד (שני). על קבוצה זו נמנות ההוראות הבאות :

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

הסבר דוגמא	דוגמא	הסבר פעולה	פקודה
$r2 \leftarrow \text{not } r2$	not r2	הפיכת ערכי הסיביות באופרנד (כל סיבית שערכה 0 תהפוך ל-1 וההיפך: 1 ל-0). האופרנד יכול להיות אוגר בלבד. אין השפעה על הדגלים.	not
$r2 \leftarrow 0$	clr r2	אפס את תוכן האופרנד.	clr
$r2 \leftarrow r2 + 1$	inc r2	הגדלת תוכן האופרנד באחד.	inc
$C \leftarrow C - 1$	dec C	הקטנת תוכן האופרנד באחד.	dec
$PC \leftarrow \text{LINE}$	jmp LINE	קפיצה בלתי מותנית אל המען	jmp

		המיוצג על ידי האופרנד.	
<p>bne</p> <p>אם ערך הדגל Z באוגר הסטטוס (PSW) הינו 0 אזי : $PC \leftarrow LINE$</p>	bne LINE	<p>bne</p> <p>הינו ראשי תיבות של : .branch if not equal (to zero) זוהי פקודת הסתעפות מותנית. הערך במצביע התוכנית (PC) יקבל את ערך האופרנד היעד אם ערכו של דגלהאפס (דגל Z) באוגר הסטטוס (PSW) הינו 0.</p>	
<p>red</p> <p>קוד ה-ascii של התו, הנקרא מלוח המקשים, יוכנס לאוגר r1.</p>	red r1	<p>red</p> <p>קריאה של תו מתוך לוח המקשים אל האופרנד.</p>	
<p>prn</p> <p>התו אשר קוד ה-ascii שלו נמצא באוגר r1 יודפס לקובץ הקלט הסטנדרטי.</p>	prn r1	<p>prn</p> <p>הדפסת התו שערך ה-ascii שלו נמצא באופרנד, אל קובץ הפלט הסטנדרטי (stdout).</p>	
<p>jsr</p> <p>$stack[SP] \leftarrow PC$ $SP \leftarrow SP - 1$ $PC \leftarrow FUNC$</p>	jsr FUNC	<p>jsr</p> <p>קריאה לשגרה (סברוטניה). הכנסת מצביע התוכנית (PC) לתוך המחסנית של זמן ריצה והעברת ערך האופרנד למצביע התוכנית (PC).</p>	

קבוצת ההוראות השלישית:

מכילה את ההוראות ללא אופרנדים – כלומר ההוראות המורכבות ממילה אחת בלבד.

ההוראות השייכות לקבוצה זו הן: rts, stop.

פקודה	הסבר פעולה	דוגמא	הסבר דוגמא
rts	הוראת חזרה משיגרה. ביצוע הוראת pop על המחסנית של זמן ריצה, והעברת הערך שהיה שם לאוגר התוכנית (PC). הוראה זו מורכבת ממילה אחת בלבד. במילה זו החלק המשמעותי הן הסיביות 6-9 המהוות את קוד ההוראה. לשאר הסיביות אין כל חשיבות.	rts	$SP \leftarrow SP + 1$ $PC \leftarrow stack[SP]$
stop	הוראה לעצירת התוכנית. ההוראה מורכבת ממילה אחת בלבד. במילה זו החלק המשמעותי הן הסיביות 6-9 המהוות את קוד ההוראה. לשאר הסיביות אין כל חשיבות.	stop	עצירת התוכנית.

מספר נקודות נוספות לגבי תיאור שפת האסמבלי:

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

ישנם ארבעה סוגי משפטים בשפת האסמבלי, והם:

סוג המשפט	הסבר כללי
משפט ריק	זוהי שורה המכילה בתוכה אך ורק תווים לבנים (white spaces) כלומר מורכבת מצירוף של 't' ו-' ' (סימני tab ורווח).
משפט הערה	זהו משפט אשר התו בעמודה הראשונה בשורה בה הוא מופיע הינו תו ';' (נקודה פסיק). על האסמבלר להתעלם לחלוטין משורה זו.
משפט הנחיה	זהו משפט המנחה את האסמבלר בעת הביצוע. יש מספר סוגי משפטי הנחיה. משפט הנחיה אינו מייצר קוד.
משפט פעולה	זהו משפט המייצר קוד. הוא מכיל בתוכו 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 בעלת תחביר חוקי. התחביר של תווית חוקית יתואר בהמשך).
התווית היא אופציונלית. לאחר מכן מופיע התו ' (נקודה) ובצמוד אליה שם ההנחיה. לאחר שם ההנחיה יופיעו (באותה שורה) הפרמטרים שלו (מספר הפרמטרים נקבע בהתאם לסוג ההנחיה).

ישנם ארבעה סוגי משפטי הנחיה והם:

1. 'data'.

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

9, 17, -57, +7 data.

יש לשים לב שהפסיקים אינם חייבים להיות צמודים למספרים. בין מספר לפסיק ובין פסיק למספר יכול להופיע מספר כלשהו של רווחים לבנים, או ללא רווחים לבנים כלל. אולם, הפסיק חייב להופיע בין הערכים.

משפט ההנחיה: 'data' מדריכה את האסמבלר להקצות מקום בהמשך חלק תמונת הנתונים (data image) שלו, אשר בו יאוחסנו הערכים המתאימים, ולקדם את מונה הנתונים, בהתאם למספר הערכים ברשימה. אם להוראת data הייתה 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.

יש לשים לב: למשפט הנחיה לא מצורפות זוג סיביות A,R,E והקידוד ממלא את כל 15 הסיביות שיש בתא זיכרון.

כלומר אם נכתוב:

XYZ:	data.	+7, -57, 17, 9
	mov	XYZ, r1

אזי יוכנס לאוגר r1 הערך +7.

לעומת זאת:

lea XYZ, r1

יכניס לאוגר r1 את המען בזיכרון (בחלק הנתונים) אשר בו אוחסן הערך +7.

2. 'string'.

הפרמטר של ההנחיה 'string' הינו מחרוזת חוקית אחת. משמעותה דומה להוראת 'data'. תווי ascii המרכיבים את המחרוזת מקודדים לפי ערכי ה-ascii המתאימים ומוכנסים אל תמונת הנתונים, לפי סדרם. בסוף יוכנס ערך אפס, לסמן סיום מחרוזת. ערך מונה הנתונים יוגדל בהתאם לאורך המחרוזת +1. אם יש תווית באותה שורה, אזי ערכה יצביע אל המקום בזיכרון, שבו מאוחסן קוד ה-ascii של התו הראשון במחרוזת, באותה צורה כפי שנעשה הדבר עבור 'data'.

כלומר משפט ההנחיה :

“abcdef” .string ABC:

מקצה “מערך תווי” באורך של 7 מילים החל מהמען המזוהה עם התווית ABC, ומאתחלת “מערך” זה לערכי ה-ascii של התווים a,b,c,d,e,f בהתאמה ולאחריהם ערך 0 לסימון סוף מחרוזת תווית.

3. ‘.entry’

להנחייה ‘.entry’ פרמטר אחד והוא שם של תווית המוגדרת בקובץ (מקבלת את ערכה שם). מטרת entry היא להצהיר על התווית הזו כעל תווית אשר קטעי אסמבלי, הנמצאים בקבצים אחרים, מתייחסים אליה.

לדוגמא :

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

מודיע שקטע (או קטעי) אסמבלי אחר, הנמצא בקובץ אחר, יתייחס לתווית HELLO.

הערה : תווית בתחילת שורת entry. חסרת משמעות והאסמבלר מתעלם מתווית זו (אפשר להוציא הודעת אזהרה).

4. ‘.extern’

להנחייה ‘.extern’ פרמטר אחד והוא שם של תווית. מטרת ההוראה היא להצהיר כי התווית מוגדרת בקובץ אחר וכי קטע האסמבלי, בקובץ זה, עושה בו שימוש. בזמן הקישור (link) תתבצע ההתאמה, בין הערך, כפי שהוא מופיע בקובץ שהגדיר את התווית, לבין ההוראות המשתמשות בו, בקבצים אחרים. גם בהוראה זו, תווית המופיעה בתחילת השורה הינה חסרת משמעות והאסמבלר מתעלם מתווית זו (אפשר להוציא הודעת אזהרה).

לדוגמא, משפט הנחית ה-‘extern’. המתאים למשפט הנחית ה-‘entry’ בדוגמא הקודמת תהיה :

HELLO .extern

שורת הוראה :

שורת הוראה מורכבת מ :

1. תווית אופציונלית.
2. שם ההוראה עצמה.
3. 0, 1 או 2 אופרנדים בהתאם להוראה.

אורכה 80 תווים לכל היותר.

שם ההוראה נכתב באותיות קטנות (lower case) והיא אחת מבין 16 ההוראות שהוזכרו לעיל.

לאחר שם ההוראה, יכול להופיע האופרנד או האופרנדים.

במקרה של שני אופרנדים, שני האופרנדים מופרדים בתו ' ', (פסיק), כקודם, לא חייבת להיות שום הצמדה של האופרנדים לתו המפריד או להוראה באופן כלשהו. כל עוד מופיעים רווחים או tabs בין האופרנדים לפסיק ובין האופרנדים להוראה, הדבר חוקי.

להוראה בעלת שני אופרנדים המבנה של :

אופרנד יעד, אופרנד מקור הוראה תווית אופציונלית
לדוגמא :

HELLO: add r7, B

לפקודה בעלת אופרנד אחד המבנה הבא :

אופרנד הוראה תווית אופציונלית
לדוגמא :

HELLO: bne XYZ

להוראה ללא אופרנדים המבנה הבא :

הוראה תווית אופציונלית
לדוגמא :

END: stop

אם מופיעה תווית בשורת ההוראה אזי היא תוכנס אל טבלת הסמלים. ערך התווית יצביע על מקום ההוראה בתוך תמונת הקוד שבונה האסמבלר.

תווית:

תווית חוקית מתחילה באות (גדולה או קטנה) ולאחריה סדרה כלשהי של אותיות וספרות, שאורכה קטן או שווה 30 תווים. התווית מסתיימת על ידי התו ' ': (נקודתיים). תו זה אינו מהווה חלק משם התווית. זהו רק סימן המייצג את סופה. כמו כן התווית חייבת להתחיל בעמודה הראשונה בשורה. אסור שיופיעו שתי הגדרות שונות לאותה התווית. התווית שלהלן הן תוויות חוקיות.

hEllo:

x:

He78902:

שם של הוראה או שם חוקי של רגיסטר אינם יכולים לשמש כשם של תווית.

התווית מקבלת את ערכה בהתאם להקשר בו היא מופיעה. תווית בהוראות 'data', 'string' תקבל את ערך מונה הנתונים (data counter) המתאים, בעוד שתווית המופיעה בשורת הוראה, תקבל את ערך מונה ההוראות (instruction counter) המתאים.

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 76, -5, +123 הינם מספרים חוקיים (אין טיפול במספרים ממשיים).

מחרוזת:

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

אסמבלר שני מעברים

כאשר מקבל האסמבלר קוד לתרגום, עליו לבצע שתי משימות עיקריות: הראשונה היא לזהות ולתרגם את קוד ההוראה, והשנייה היא לקבוע מענים לכל המשתנים והנתונים המופיעים בתוכנית. לדוגמא: אם האסמבלר קורא את קטע הקוד הבא:

```

MAIN:      mov     r5[r2],LENGTH
           add     r2,STR
LOOP:      jmp     END
           prn     #-5
           sub     r1, r4
           inc     K

           mov     r7[r6],r3
           bne     LOOP
END:       stop
STR:       .string "abcdef"
LENGTH:    .data   6,-9,15
K:         .data   22

```

עליו להחליף את שמות הפעולה mov, add, jmp, prn, sub, inc, bne, stop בקוד הבינארי השקול להם במחשב שהגדרנו.

כמו כן, על האסמבלר להחליף את הסמלים K,STR, LENGTH, MAIN, LOOP, END במענים של האתרים שהוקצו לנתונים, ובמענים של ההוראות המתאימות.

נניח שקטע הקוד למעלה יאוחסן (הוראות ונתונים) בקטע זיכרון החל ממען 0100 (בבסיס 10) בזיכרון. במקרה זה נקבל את ה"תרגום" הבא:

לתשומת לב: המקפים המופיעים בקידוד הבינארי הם רק לצורך הפרדה של השדות השונים בקידוד ונועדו לשם המחשה בלבד.

Label	Decimal Address	Base 16 Address	Command	Operands	Binary machine code
MAIN:	0100	64	mov	r5[r2],LENGTH	111-10-0000-10-01-00
	0101	65		קידוד האוגרים	0-000010-000101-00
	0102	66		כתובת של LENGTH	0000001111111-10
	0103	67	add	r2, STR	111-10-0010-11-01-00
	0104	68		קידוד מספר האוגר	0-000010-000000-00
	0105	69		כתובת של STR	0000001111000-10
LOOP:	0106	6A	jmp	END	111-01-1001-00-01-00
	0107	6B		כתובת של END	0000001110111-10
	0108	6C	prn	#-5	111-01-1100-00-00-00
	0109	6D		המספר -5	1111111111011-00
	0110	6E	sub	r1,r4	111-10-0011-11-11-00
	0111	6F		קידודי מספרי האוגרים	0-000001-000100-00
	0112	70	inc	K	111-01-0111-00-01-00
	0113	71		כתובת של K	0000010000010-10
	0114	72	mov	r7[r6],r3	111-10-0000-10-11-00
	0115	73		קידוד האוגרים של אופרנד המקור	0-000110-000111-00

	0116	74		קידוד מספר האוגר	0-000000-000011-00
	0117	75	bne	LOOP	111-01-1010-00-01-00
	0118	76		כתובת של LOOP	0000001101010-10
END:	0119	77	stop		111-00-1111-00-00-00
STR:	0120	78	.string	"abcdef"	000000001100001
	0121	79			000000001100010
	0122	7A			000000001100011
	0123	7B			000000001100100
	0124	7C			000000001100101
	0125	7D			000000001100110
	0126	7E			0000000000000000
LENGTH:	0127	7F	.data	6,-9,15	000000000000110
	0128	80			11111111110111
	0129	81			000000000001111
K:	0130	82	.data	22	000000000010110

אם האסמבלר מחזיק טבלה שבה רשומים כל שמות הפעולה של ההוראות והקודים הבינאריים המתאימים להם, אזי שמות הפעולה ניתנים להמרה בקלות. כאשר נקרא שם פעולה, אפשר פשוט לעיין בטבלה ולמצוא את הקוד הבינארי השקול.

כדי לעשות את אותה פעולה לגבי מענים סמליים, יש צורך לבנות טבלה דומה. אולם הקודים של הפעולות ידועים מראש, ואילו היחס בין הסמלים, שבשימוש המתכנת, לבין מעני התווית שלהם בזיכרון, אינו ידוע, עד אשר התוכנית מקודדת ונקראת על יד המחשב.

בדוגמא שלפנינו, אין האסמבלר יכול לדעת שהסמל LOOP משויך למען 0106 (עשרוני), אלא רק לאחר שהתוכנית נקראה כולה.

לכן מפרידים את הטיפול בסמלים לשני שלבים. בשלב הראשון בונים טבלה של כל הסמלים והערכים המספריים המשויכים להם, ובשלב השני מחליפים את כל הסמלים, המופיעים בפקודות התוכנית בשדה המען, בערכיהם המספריים. הביצוע של שני שלבים אלה כרוך בשתי סריקות (הנקראות "מעברים") של קובץ המקור.

במעבר הראשון נבנית טבלת סמלים בזיכרון, שמותאמים בה מענים לכל הסמלים. בדוגמא דלעיל, טבלת הסמלים לאחר מעבר ראשון היא:

סמל	ערך דצימלי
MAIN	100
LOOP	106
END	119
STR	120
LENGTH	127
K	130

במעבר השני נעשית ההחלפה, כדי לתרגם את הקוד לבינארי. בתחילת המעבר השני צריכים הערכים של כל הסמלים להיות כבר ידועים.

לתשומת לב: האסמבלר, על שני המעברים שלו, נועד כדי לתרגם את תכנית המקור לתכנית בשפת מכונה. בזמן פעולת האסמבלר, התכנית טרם מוכנה לטעינה לזיכרון לצורך ביצוע. לאחר השלמת תהליך התרגום, קוד המכונה יכול לעבור לשלב הקישור/טעינה ולאחר מכן לשלב הביצוע.

המעבר הראשון

במעבר הראשון נדרשים כללים כדי לקבוע איזה ערך מען ישוּיך לכל סמל. העיקרון הבסיסי הוא לספור את המקומות בזיכרון, שאותם צורכת כל הוראה כאשר היא נקראת. אם כל הוראה תיטען לאחר האסמבלר, לאתר העוקב של אתר ההוראה הקודמת, תציין ספירה כזאת את מען ההוראה. הספירה נעשית על ידי האסמבלר ומוחזקת באוגר הנקרא מונה אתרים (IC). ערכו ההתחלתי הוא 0, ולכן נטען משפט ההוראה הראשון במען 0. הוא משתנה על ידי כל הוראה, או הוראה מדומה, המקצה מקום. לאחר שהאסמבלר קובע מהו אורך ההוראה, תוכנו של מונה האתרים עולה במספר הבתים, הנתפסים על ידי ההוראה, וכך הוא מצביע על התא הריק הבא.

כאמור, כדי לקודד את ההוראות בשפת מכונה, מחזיק האסמבלר טבלה, שיש בה קוד מתאים לכל הוראה. בזמן התרגום מחליף האסמבלר כל הוראה בקוד שלה. אך פעולת ההחלפה אינה כה פשוטה. יש הרבה הוראות המשתמשות בצורות מיעון שונות. אותה הוראה יכולה לקבל משמעויות שונות, בכל אחת מצורות המיעון, ולכן יתאימו לה קודים שונים. לדוגמה, הוראת ההזזה mov יכולה להתייחס להעברת תוכן תא זיכרון לאוגר, או להעברת תוכן אוגר לאוגר, וכן הלאה. לכל צורה כזאת של mov מתאים קוד שונה.

על האסמבלר לסרוק את שורת ההוראה בשלמותה, ולהחליט לגבי קוד ההוראה לפי האופרנדים שלה. בדרך כלל מתחלק קוד ההוראה המתורגם לשדה קוד ההוראה ושדות נוספים, המכילים מידע לגבי שיטות המיעון.

במחשב שלנו קיימת גמישות לגבי שיטת המיעון של שני האופרנדים. הערה: דבר זה לא מחייב לגבי כל מחשב. ישנם מחשבים בהם כל הפקודות הן בעלות אופרנד יחיד (והפעולות מתבצעות על אופרנד זה ואוגר קבוע) או מחשבים המאפשרים מבחר של שיטות מיעון עבור אופרנד אחד והאופרנד השני חייב להיות אוגר כלשהו, או מחשבים בעלי 3 אופרנדים, כאשר האופרנד השלישי משמש לאחסון תוצאת הפעולה.

כאשר נתקל האסמבלר בתווית המופיעה בתחילת השורה, הוא יודע שלפניו הגדרה של תווית, ואז ניתן לה מען – תוכנו הנוכחי של מונה האתרים. כך מקבלות כל התוויות את מעניהן בעת ההגדרה. תוויות אלה מוכנסות לטבלת הסמלים, המכילה בנוסף לשם התווית גם את מענה ומאפיינים נוספים שלה, כגון סוג התווית. כאשר תהיה התייחסות לתווית כזאת בשדה המען של ההוראה, יוכל האסמבלר לשלוף את המען המתאים מהטבלה.

כידוע, מתכנת יכול להתייחס גם לסמל, שלא הוגדר עד כה בתכנית, אלא רק לאחר מכן. לדוגמה: פקודת הסתעפות למען, המופיע בהמשך הקוד:

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

כאשר מגיע האסמבלר לשורה זו (bne A), הוא עדיין לא הקצה מען לתווית A ולכן אינו יכול להחליף את הסמל A במענו בזיכרון. נראה בהמשך כיצד נפתרת בעיה זו.

בנוסף, ניתן לייצר במעבר הראשון את הקוד של המילה הראשונה של כל פקודה ואת קוד המכונה של כל המילים ב-data.

המעבר השני

בעת המעבר הראשון, אין האסמבלר יכול להשלים בטבלה את מעני הסמלים שלא הוגדרו עדיין, והוא מציין אותם באפסים. רק לאחר שהאסמבלר עבר על כל התכנית, כך שכל התוויות הוכנסו כבר לטבלת הסמלים, יכול האסמבלר להחליף את התוויות, המופיעות בשדה המען של ההוראה, במעניהן המתאימים. לשם כך עובר האסמבלר שנית על כל התוכנית, ומחליף את התוויות, המופיעות בשדה המען, במעניהן המתאימים מתוך הטבלה. זהו המעבר השני, ובסופו תהיה התוכנית מתורגמת בשלמותה.

הפרדת הוראות ונתונים

בתכנית בקוד המכונה מבחינים בשני סוגים של תוכן: הוראות ונתונים. יש לארגן את הקוד כך שתהיה אבחנה 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 הקוד מהנתונים היא, שלפעמים עלול המעבד, בעקבות שגיאה לוגית קלה בתכנית, לנסות "לבצע" את הנתונים כאילו היו הוראות חוקיות. למשל, שגיאה שיכולה לגרום זאת היא השמטת הוראת עצירה או הסתעפות לא נכונה. התכנית כמובן לא תעבוד נכון, אך לרוב הנזק הוא יותר חמור, כי נוצרת חריגת חומרה ברגע שמתבצעת הוראה שאינה חוקית.

בתוכנית האסמבלר, שעליכם לממש, יש להפריד בין קטע הנתונים לקטע ההוראות. **כלומר בקבצי הפלט תהיה הפרדה של הוראות ונתונים לשני קטעים נפרדים, ואילו בקובץ הקלט אין חובה שתהיה הפרדה.** בהמשך מתואר אלגוריתם של האסמבלר, ובו פרטים כיצד לבצע את ההפרדה.

גילוי שגיאות בתכנית המקור

האסמבלר יכול לגלות ולדווח על שגיאות בתחביר של השפה, כגון הוראה שאינה קיימת, מספר אופרנדים שגוי, אופרנד שאינו מתאים להוראה ועוד. כמו כן מוודא האסמבלר שכל הסמלים מוגדרים פעם אחת בדיוק. מכאן שאת השגיאות, המתגלות בידי האסמבלר, אפשר לשייך בדרך כלל לשורת קלט מסוימת.

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

הטבלה הבאה מכילה מידע על שיטות מיעון חוקיות, עבור אופרנד המקור, ואופרנד היעד, עבור הפקודות השונות הקיימות בשפה הנתונה :

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

שגיאות נוספות אפשריות הן פקודה לא חוקית, שם רגיסטר לא חוקי, תווית לא חוקית וכו'.

אלגוריתם של האסמבלר

להלן נציג אלגוריתם שלדי למעבר הראשון ולמעבר השני : אנו נניח כי קוד המכונה מחולק לשני אזורים, אזור ההוראות (code) ואזור הנתונים (data) . לכל אזור יש מונה משלו, ונסמנם באותיות IC (מונה ההוראות - Instruction-counter) ו-DC (מונה הנתונים - Data-counter). האות L תסמן את מספר המילים שתופסת ההוראה.

מעבר ראשון

1. $DC \leftarrow 0, IC \leftarrow 0$.
2. קרא שורה.
3. האם השדה הראשון הוא סמל? אם לא, עבור ל-5.
4. הצב דגל "יש סמל".
5. האם זוהי הנחיה לאחסון נתונים, כלומר, האם הנחית data או string? אם לא, עבור ל-8.
6. אם יש סמל, 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, אחסן אותם בזיכרון, עדכן את מונה הנתונים בהתאם לאורכם, חזור ל-2.
8. האם זו הנחית extern או הנחית entry? אם לא, עבור ל-11.
9. האם זוהי הצהרת extern? אם כן, הכנס את הסמלים לטבלת הסמלים החיצוניים, ללא מען.
10. חזור ל-2.
11. אם יש סמל, הכנס אותו לטבלת הסמלים עם סימון (סמל מסוג code). ערכו יהיה IC (אם הסמל כבר נמצא בטבלה יש להודיע על שגיאה).
12. חפש בטבלת ההוראות, אם לא מצאת – הודע על שגיאה בקוד ההוראה.
13. נתח את מבנה האופרנדים של ההוראה וחשב את L. ניתן לייצר כבר כאן את קוד המילה הראשונה של הפקודה
14. $IC \leftarrow L + IC$
15. חזור ל-2.

מעבר שני

1. $IC \leftarrow 0$
2. קרא שורה. אם סיימת, עבור ל-11.
3. אם השדה הראשון הוא סמל, דלג עליו.
4. האם זוהי הוראה מדומה (.string, .data)? אם כן, חזור ל-2.
5. האם זוהי הנחיה (.extern, .entry)? אם לא, עבור ל-7.
6. זהה את ההנחיה. השלם את הפעולה המתאימה לה. אם זאת הנחיית entry. סמן את הסמלים המתאימים כ-entry. חזור ל-2.
7. הערך את האופרנדים, חפש בטבלת ההוראות, החלף את ההוראה בקוד המתאים.
8. אחסן את האופרנדים החל מהבית הבא. אם זהו סמל, מצא את המען בטבלת הסמלים, חשב מענים, קודד שיטת מיעון.
9. $IC \leftarrow IC + L$
10. חזור ל-2.
11. שמור על קובץ נפרד את אורך התוכנית, אורך הנתונים, טבלת סמלים חיצוניים, טבלת סמלים עם סימוני נקודות כניסה.

נפעיל אלגוריתם זה על תוכנית הדוגמא שראינו קודם :

```

MAIN:      mov    r5[r2],LENGTH
           add    r2,STR
LOOP:      jmp    END
           prn    #-5
           sub    r1, r4
           inc    K

           mov    r7[r6],r3
           bne    LOOP
END:       stop
STR:       .string "abcdef"
LENGTH:   .data   6,-9,15
K:         .data   22

```

נבצע עתה מעבר ראשון על הקוד הנתון. נבצע במעבר זה גם את החלפת ההוראה בקוד שלה. כמו כן נבנה את טבלת הסמלים. את החלקים שעדיין לא מתורגמים בשלב זה, נשאיר כמות שהם. נניח שהקוד ייטען החל מהמען 100 (בבסיס 10).

Label	Decimal Address	Base 16 Address	Command	Operands	Binary machine code
MAIN:	0100 0101 0102	64 65 66	mov	r5[r2],LENGTH קידוד האוגרים כתובת של LENGTH	111-10-0000-10-01-00 0-000010-000101-00 ?
	0103 0104 0105	67 68 69	add	r2, STR קידוד מספר האוגר כתובת של STR	111-10-0010-11-01-00 0-000010-000000-00 ?
LOOP:	0106 0107	6A 6B	jmp	END כתובת של END	111-01-1001-00-01-00 ?
	0108 0109	6C 6D	prn	#-5 המספר -5	111-01-1100-00-00-00 1111111111011-00
	0110 0111	6E 6F	sub	r1,r4 קידודי מספרי האוגרים	111-10-0011-11-11-00 0-000001-000100-00
	0112 0113	70 71	inc	K כתובת של K	111-01-0111-00-01-00 ?
	0114 0115 0116	72 73 74	mov	r7[r6],r3 קידוד האוגרים של אופרנד המקור קידוד מספר האוגר	111-10-0000-10-11-00 0-000110-000111-00 0-000000-000011-00
	0117 0118	75 76	bne	LOOP כתובת של LOOP	111-01-1010-00-01-00 ?
END:	0119	77	stop		111-00-1111-00-00-00
STR:	0120	78	.string	"abcdef"	000000001100001
	0121	79			000000001100010
	0122	7A			000000001100011
	0123	7B			000000001100100
	0124	7C			000000001100101
	0125	7D			000000001100110
	0126	7E			000000000000000
LENGTH:	0127 0128 0129	7F 80 81	.data	6,-9,15	000000000000110 11111111110111 000000000001111
K:	0130	82	.data	22	000000000010110

טבלת הסמלים :

סמל	ערך דצימלי
MAIN	100
LOOP	106
END	119
STR	120
LENGTH	127
K	130

נבצע עתה את המעבר השני ונרשום את הקוד בצורתו הסופית :

Label	Decimal Address	Base 16 Address	Command	Operands	Binary machine code
MAIN:	0100	64	mov	r5[r2],LENGTH	111-10-0000-10-01-00
	0101	65		קידוד האוגרים	0-000010-000101-00
	0102	66		כתובת של LENGTH	000000111111-10
	0103	67	add	r2, STR	111-10-0010-11-01-00
	0104	68		קידוד מספר האוגר	0-000010-000000-00
	0105	69		כתובת של STR	0000001111000-10
LOOP:	0106	6A	jmp	END	111-01-1001-00-01-00
	0107	6B		כתובת של END	0000001110111-10
	0108	6C	prn	#-5	111-01-1100-00-00-00
	0109	6D		המספר -5	1111111111011-00
	0110	6E	sub	r1,r4	111-10-0011-11-11-00
	0111	6F		קידודי מספרי האוגרים	0-000001-000100-00
	0112	70	inc	K	111-01-0111-00-01-00
	0113	71		כתובת של K	0000010000010-10
	0114	72	mov	r7[r6],r3	111-10-0000-10-11-00
	0115	73		קידוד האוגרים של אופרנד המקור	0-000110-000111-00
	0116	74		קידוד מספר האוגר	0-000000-000011-00
	0117	75	bne	LOOP	111-01-1010-00-01-00
	0118	76		כתובת של LOOP	0000001101010-10
END:	0119	77	stop		111-00-1111-00-00-00
STR:	0120	78	.string	"abcdef"	000000001100001
	0121	79			000000001100010
	0122	7A			000000001100011
	0123	7B			000000001100100
	0124	7C			000000001100101
	0125	7D			000000001100110
	0126	7E			000000000000000
LENGTH:	0127	7F	.data	6,-9,15	000000000000110
	0128	80			111111111110111
	0129	81			000000000001111
K:	0130	82	.data	22	000000000010110

לאחר סיום עבודת האסמבלר, קוד המכונה מועבר לשלבי הקישור והטעינה.

לא נדון כאן באופן עבודת שלבי הקישור/טעינה (כאמור, אלה אינם למימוש בפרויקט זה).
לאחר סיום שלבים אלה, התוכנית טעונה בזיכרון ומוכנה לריצה.

קבצי קלט ופלט של האסמבלר

על תוכנית האסמבלר לקבל כארגומנטים של שורת פקודה (command line arguments) רשימה של קבצי מקור בשפת אסמבלי. אלו הם קבצי טקסט, ובהם רשומות הוראות בתחביר של שפת האסמבלי, שהוגדרה למעלה. עבור כל קובץ מקור יוצר האסמבלר קובץ מטרה (object). כמו כן ייווצר (עבור כל קובץ מקור) קובץ externals, באם המקור הצהיר על משתנים חיצוניים, וקובץ entries, באם המקור הצהיר על משתנים מסיימים כעל נקודות כניסה.

קבצי המקור חייבים להיות בעלי הסיומת ".as". השמות x.as, y.as, ו-hello.as הם שמות חוקיים. הפעלת האסמבלר על הקבצים הללו נעשית ללא ציון הסיומת. לדוגמא: אם תוכנית האסמבלר שלנו נקראת assembler, אזי שורת הפקודה הבאה:

```
assembler x y hello
```

תגרום לכך שתוכנית האסמבלר שלנו תפעל על הקבצים: x.as, y.as, hello.as.

האסמבלר יוצר את שמות קבצי ה-object, קבצי ה-entries וקבצי ה-externals על ידי לקיחת שם הקובץ כפי שהופיע בשורת ההוראה והוספת הסיומת ".ob". עבור קובץ ה-object, סיומת ".ent". עבור קובץ ה-entries, וסיומת ".ext". עבור קובץ ה-externals.

מבנה כל קובץ פלט יתואר בהמשך.

לדוגמא: הפקודה: assembler x : ואת הקבצים x.ent ו-x.ext אם קיימים entries/externals בקובץ. תיצור את הקובץ x.ob

אופן פעולת האסמבלר

העבודה על קובץ מקור נתון נעשית בצורה הבאה:

האסמבלר מחזיק שני מערכים, שייקראו להלן מערך הקוד ומערך הנתונים. מערכים אלו נותנים למעשה תמונה של זיכרון המכונה (גודל כל כניסה במערך זהה לגודלה של מילת מכונה: 15 סיביות). במערך הקוד מכניס האסמבלר את הקידוד של הוראות המכונה בהן הוא נתקל במהלך המעבר על קובץ המקור. במערך הנתונים מכניס האסמבלר את קידוד הנתונים שנקראו מקובץ המקור (שורות מסוג data ו-string).

לאסמבלר יש שני מונים: מונה ההוראות (IC) ומונה הנתונים (DC). מונים אלו מצביעים על המקום הבא הפנוי במערכים לעיל בהתאמה. כשמתחיל האסמבלר את פעולתו על קובץ מסוים 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, פעולה, הנחיה או שורה ריקה) ופועל בהתאם.

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

2. שורת פעולה:

כאשר האסמבלר נתקל בשורת פעולה הוא מחליט מהי הפעולה, מהי שיטת המיעון ומי הם האופרנדים. (מספר האופרנדים אותם הוא מחפש נקבע בהתאם לפעולה אותה הוא מצא). האסמבלר קובע לכל אופרנד את ערכו בצורה הבאה:

- אם זה אוגר – ערכו של האופרנד הוא מספר האוגר.
- אם זו תווית (מיעון ישיר) – ערכו של האופרנד הוא ערך התווית כפי שמופיע בטבלת הסמלים.

- אם זה מספר (מיעון מידי) – ערכו הוא המספר עצמו.
 - אם זו שיטת מיעון אחרת, ערכו של האופרנד נקבע לפי המפרט של שיטת המיעון (ראה תאור שיטות המיעון לעיל)
- קביעת שיטת המיעון נעשית בהתאם לתחביר של האופרנד, כפי שהוא מתואר בחלק העוסק בשיטות המיעון. למשל, התו # מציין מיעון מידי, תווית מציינת מיעון ישיר, שם של אוגר מציין מיעון אוגר.
- שימו לב: ערך שדה האופרנד הינו ערך תווית המשתנה, כפי שהוא מופיע בטבלת הסמלים.
- לאחר שהאסמבלר החליט לגבי הנ"ל (פעולה, שיטת מיעון אופרנד מקור, שיטת מיעון אופרנד יעד, אוגר אופרנד מקור, אוגר אופרנד יעד, האם נדרשת מילה נוספת עבור אופרנד מקור באם יש, האם נדרשת מילה נוספת עבור אופרנד יעד באם יש) הוא פועל באופן הבא:
- אם זוהי הוראה בעלת שני אופרנדים, אזי האסמבלר מכניס למערך הקוד, במקום עליו מצביע מונה ההוראות, מספר אשר ייצג (בשיטת הייצוג של הוראות המכונה כפי שתוארו קודם לכן) את קוד הפעולה, שיטות המיעון, ואת המידע על האוגרים. בנוסף הוא "משריין" מקום עבור מספר המילים הנוספות, הנדרשות עבור פקודה זו ומגדיל את מונה הקוד בהתאם.
- אם ההוראה היא בעלת אופרנד אחד בלבד, כלומר האופרנד הראשון (אופרנד המקור) אינו מופיע, אזי התרגום הינו זהה לחלוטין, למעט העובדה שסיביות מיעון המקור במלה הראשונה המוכנסת לזיכרון (האמורות לייצג את המידע על אופרנד המקור) יכולות להיות בעלות כל ערך אפשרי מכיוון שערך זה אינו משמש כלל את ה-CPU.
- אם ההוראה היא ללא אופרנדים (rts, stop) אזי למקום במערך הקוד, שאליו מצביע מונה ההוראות, יוכנס מספר אשר מקודד אך ורק את קוד ההוראה של הפעולה. שיטות המיעון ומידע על האוגרים של אופרנדי המקור והיעד, יכולים להיות בעלי ערך כלשהו ללא הגבלה.
- אם לשורת הקוד קיימת תווית, אזי התווית מוכנסת אל טבלת הסמלים תחת השם המתאים, ערכה הוא ערך מונה ההוראות לפני קידוד ההוראה. טיפוסה הוא relocatable.
3. שורת הנחיה:
- כאשר האסמבלר נתקל בהנחיה הוא פועל בהתאם לסוג שלה, באופן הבא:
- I. 'data'.
- האסמבלר קורא את רשימת המספרים, המופיעה לאחר 'data'. הוא מכניס כל מספר שנקרא אל מערך הנתונים ומקדם את מצביע הנתונים באחד, עבור כל מספר שהוכנס.
- אם ל-'data' יש תווית לפניה, אזי תווית זו מוכנסת לטבלת הסמלים. היא מקבלת את הערך של מונה הנתונים, לפני שהנתונים הוכנסו אל תוך הקוד + אורך הקוד הכללי. הטיפוס שלה הוא relocatable, והגדרתה ניתנה בחלק הנתונים.
- II. 'string'.
- ההתנהגות לגבי 'string' דומה לזו של 'data'. אלא שקודי ה-ascii של התווים הנקראים הם אלו המוכנסים אל מערך הנתונים. לאחר מכן מוכנס הערך 0 (אפס, המציין סוף מחרוזת) אל מערך הנתונים. מונה הנתונים מקודם באורך המחרוזת + 1 (כי גם האפס תופס מקום). ההתנהגות לגבי תווית המופיעה בשורה, זהה להתנהגות במקרה של 'data'.
- III. 'entry'.
- זוהי בקשה מן האסמבלר להכניס את התווית המופיעה לאחר 'entry'. אל קובץ ה-entries. האסמבלר רושם את הבקשה ובסיום העבודה, התווית הנ"ל תירשם בקובץ ה-entries. 'entry' באה להכריז על תווית שנעשה בה שימוש בקובץ אחר, וכי על תוכנית הקישור להשתמש במידע המצוי בקובץ ה-entries ובקובץ ה-externals כדי להתאים בין ההתייחסויות ל-externals.

IV. 'extern'

זוהי בקשה הבאה להצהיר על משתנה המוגדר בקובץ אחר, אשר קטע האסמבלי בקובץ עכשווי עושה בו שימוש.

האסמבלר מכניס את המשתנה המבוקש אל טבלת הסמלים. ערכו הוא אפס (או כל ערך אחר), טיפוסו הוא external, היכן ניתנה הגדרתו אין יודעים (ואין זה משנה עבור האסמבלר).

יש לשים לב! בפעולה או בהנחיה אפשר להשתמש בשם של משתנה, אשר ההצהרה עליו ניתנת בהמשך הקובץ (אם באופן עקיף על ידי תווית ואם באופן מפורש על ידי 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 למען שלאחר ההוראה הראשונה (מען 101 או 102 או 103, תלוי באורך ההוראה הראשונה) וכך הלאה עד לתרגום ההוראה האחרונה. מיד לאחר קידוד ההוראה האחרונה, מכילה תמונת הזיכרון את הנתונים שנבנו על ידי הוראות 'data' ו-'string'. הנתונים שיהיו ראשונים הם הנתונים המופיעים ראשונים בקובץ האסמבלי, וכך הלאה.

התייחסות בקובץ האסמבלי למשתנה, שהוגדר בקובץ, תקודד כך שתצביע על המקום המתאים, בתמונת הזיכרון שבונה האסמבלר. עקרונית פורמט של קובץ object הינו תמונת הזיכרון הנ"ל.

קובץ object מורכב משורות שורות של טקסט, השורה הראשונה מכילה, (בבסיס 16) את אורך הקוד (במילות זיכרון) ואת אורך הנתונים (במילות זיכרון). שני המספרים מופרדים ביניהם על ידי רווח. השורות הבאות מתארות את תוכן הזיכרון (שוב, בבסיס 16)

בהמשך מופיע קובץ object לדוגמא ששמו: ps.ob המתאים ל-ps.as

בנוסף עבור כל תא זיכרון המכיל הוראה (לא data) מופיע מידע עבור תכנית הקישור. מידע זה ניתן ע"י 2 הסיביות הימניות של הקידוד (שדה ה-E,R,A)

האות 'A' (קיצור של absolute) מציינת את העובדה שתוכן התא אינו תלוי היכן בפועל יטען הקובץ של קוד המכונה בזמן ביצוע התכנית (האסמבלר יוצא מתוך הנחה שהקובץ נטען החל ממען אפס). במקרה כזה 2 הסיביות יכילו את הערך 00

האות 'R' (קיצור של relocatable) מציינת שתוכן תא הזיכרון תלוי במקום בזיכרון שבו ייטען בפועל קוד המכונה של התכנית בזמן ביצועה, ולכן יש להוסיף לתוכן התא בזמן הטעינה את ההיסט (Offset) המתאים. ה-offset הינו מען הזיכרון שבו תטען, למעשה, ההוראה, אשר האסמבלר אומר שעליה להיטען במען אפס. במקרה כזה 2 הסיביות יכילו את הערך 10

האות 'E' (קיצור של external) מציינת שתוכן תא הזיכרון תלוי בכתובתו של משתנה חיצוני (external). וכי בזמן הקישור יוכנס כאן הערך המתאים. במקרה כזה 2 הסיביות יכילו את הערך 01

פורמט קובץ ה-entries

קובץ ה-entries בנוי משורות טקסט. כל שורה מכילה שם של סמל שהוגדר כ-entry ואת ערכו, כפי שחושב בטבלת הסמלים של קובץ המקור (ראה לדוגמא את הקובץ ps.ent המתאים לקובץ האסמבלי ששמו ps.as).

פורמט קובץ ה-externals

קובץ ה-externals בנוי אף הוא משורות טקסט. כל שורה מכילה שם של סמל שהוגדר כ-external ואת כתובת המילה בקוד המכונה שבה יש התייחסות לסמל חיצוני זה (לדוגמא הקובץ ps.ext מתאים לקובץ האסמבלי ששמו ps.as).

להלן הקובץ ps.as לדוגמא :

```
; file ps.as

.entry LOOP
.entry LENGTH
.extern L3
.extern W
MAIN:      mov    r5[r2],W
           add    r2,STR
LOOP:      jmp     W
           prn    #-5
           sub    r1, r4
           inc    K

           mov    r7[r6],r3
           bne    L3
END:       stop
STR:       .string "abcdef"
LENGTH:    .data  6,-9,15
K:         .data  22
```

הקובץ שלהלן הוא קובץ object ששמו בעל סיומת '.ob'. זהו קובץ שהתקבל מהפעלת התוכנית assembler על קובץ האסמבלר דלעיל. להלן דוגמת הקידוד לביטים ולאחריה פורמט קובץ ה-OB. **כל תוכן הקובץ מיוצג על ידי מספרים בבסיס 16.**

קובץ ps.ob :

Label	Decimal Address	Base 16 Address	Command	Operands	Binary machine code
MAIN:	0100	64	mov	r5[r2],W	111-10-0000-10-01-00
	0101	65		קידוד האוגרים	0-000010-000101-00
	0102	66		כתובת של W	000000000000-01
	0103	67	add	r2, STR	111-10-0010-11-01-00
	0104	68		קידוד מספר האוגר	0-000010-000000-00
	0105	69		כתובת של STR	0000001111000-10
LOOP:	0106	6A	jmp	W	111-01-1001-00-01-00
	0107	6B		כתובת של W	000000000000-01
	0108	6C	prn	#-5	111-01-1100-00-00-00
	0109	6D		המספר -5	111111111011-00
	0110	6E	sub	r1,r4	111-10-0011-11-11-00
	0111	6F		קידודי מספרי האוגרים	0-000001-000100-00
	0112	70	inc	K	111-01-0111-00-01-00
	0113	71		כתובת של K	0000010000010-10
	0114	72	mov	r7[r6],r3	111-10-0000-10-11-00
	0115	73		קידוד האוגרים באופרנד המקור	0-000110-000111-00
	0116	74		קידוד מספר האוגר	0-000000-000011-00
	0117	75	bne	L3	111-01-1010-00-01-00
	0118	76		כתובת של L3	000000000000-01
	0119	77	stop		111-00-1111-00-00-00

<i>STR:</i>	0120	78	.string	"abcdef"	000000001100001
	0121	79			000000001100010
	0122	7A			000000001100011
	0123	7B			000000001100100
	0124	7C			000000001100101
	0125	7D			000000001100110
	0126	7E			000000000000000
<i>LENGTH:</i>	0127	7F	.data	6,-9,15	000000000000110
	0128	80			11111111110111
	0129	81			000000000001111
<i>K:</i>	0130	82	.data	22	00000000010110

כלומר תוכן קובץ ps.ob הוא:

Base 16 Address	Base 16 code
14 B	
64	7824
65	0214
66	0001
67	78B4
68	0200
69	01E2
6A	7644
6B	0001
6C	7700
6D	7FEC
6E	78FC
6F	0110
70	75C4
71	020A
72	782C
73	061C
74	000C
75	7684
76	0001
77	73C0
78	0061
79	0062
7A	0063
7B	0064
7C	0065
7D	0066
7E	0000
7F	0006
80	7FF7
81	000F
82	0016

קובץ: ps.ent

LOOP 6A
LENGTH 7F

קובץ: ps.ext

W 66
W 6B
L3 76

לתשומת לבך : אם בקובץ מסוים אין הצהרת extern. אזי לא ייווצר עבורו קובץ ext. המתאים.
כנ"ל עבור קבצים שאינם מכילים הודעות entry, במקרה זה לא ייווצר קובץ ent. מתאים.
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 1000 של הזיכרון במעבד הדמיוני). ולכן אורכה של התוכנית המתורגמת, אינו אמור להיות צפוי מראש. אולם בכדי להקל במימוש התוכנית, ניתן להניח גודל מקסימלי. לפיכך יש אפשרות להשתמש במערכים, לשם מטרה זו. במבנים אחרים בפרויקט, יש להשתמש על פי יעילות/תכונות נדרשות.
- קבצי הפלט של התוכנית, צריכים להיות בפורמט המופיע למעלה. שמם של קבצי הפלט צריך להיות תואם לשמה של תוכנית הקלט, פרט לסיומות. למשל, אם תוכנית הקלט היא prog.as אזי קבצי הפלט שייווצרו הם : prog.ob, prog.ext, prog.ent .
- אופן הרצת התוכנית צריך להיות תואם לנדרש בממ"ן, ללא שינויים כלשהם. אין להוסיף תפריטים למיניהם וכדומה. הפעלת התוכנית תיעשה רק ע"י ארגומנטים של שורת פקודה.
- יש להקפיד לחלק את התוכנית למודולים. אין לרכז מספר מטרות במודול יחיד. מומלץ לחלק למודולים כגון : מבני נתונים, מעבר ראשון, מעבר שני, טבלת סמלים וכדומה.
- יש להקפיד ולתעד את הקוד, בצורה מלאה וברורה.
- יש להקפיד על התעלמות מרווחים, ולהיות סלחנים כלפי תוכניות קלט, העושות שימוש ביותר רווחים מהנדרש. למשל, אם לפקודה ישנם שני אופרנדים המופרדים בפסיק, אזי לפני שם הפקודה או לאחריה או לאחר האופרנד הראשון או לאחר הפסיק, יכול להיות מספר רווחים כלשהו, ובכל המקרים זו צריכה להיחשב פקודה חוקית (לפחות מבחינת הרווחים).
- במקרה של תוכנית קלט, המכילה שגיאות תחביריות, נדרש להפיק, כמו באסמבלר אמיתי, את כל השגיאות הקיימות, ולא לעצור לאחר היתקלות בשגיאה הראשונה. כמובן שעבור קובץ שגוי תחבירי, אין להפיק את קבצי הפלט (ob, ext, ent) אלא רק לדווח על השגיאות שנמצאו .

תם ונשלם חלק ההסברים והגדרת הפרויקט.

בשאלות ניתן לפנות אל :

קבוצת הדיון באתר הקורס, לכל אחד מהמנחים בשעות הקבלה שלהם. להזכירכם, באפשרותו של כל סטודנט לפנות לכל מנחה, לאו דווקא למנחה הקבוצה שלו לקבלת עזרה. שוב מומלץ לכל אלה מכם שטרם בדקו את אתר הקורס, לעשות זאת. נשאלות באתר זה הרבה שאלות בנושא חומר הלימוד והממ"נים, והתשובות יכולות לעזור לכולם.

לתשומת לבכם, לא יתקבלו ממ"נים באיחור ללא סיבה מוצדקת, באישור צוות הקורס.

בהצלחה!!!!