

ה א ו נ י ב ר ס י ט ה ה פ ת ו ח ה

20465

מעבדה בתכנות מערכות

חוبراַת הקורס – אָבִיב 2021

כתבה : מיכל אבימול

פברואר 2021 – סמסטר אָבִיב – תשפ"א

פנימי – לא להפצתה.

© כל הזכויות שמורות לאוניברסיטת הפתוחה.

תוכן העניינים

א	אל הסטודנט
ג	1. לוח זמנים ופעולות
ה	2. תיאור המטלות
ז	3. התנאים לקבלת נקודות זכות
1	ממ"נ 11
5	ממ"נ 12
8	ממ"נ 22
14	ממ"נ 23
16	ממ"נ 14

אל הסטודנט,

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

לקורס קיימים אתר באינטרנט בו תמצאו חומר למידה נוספים, אותם מפרסם/מתמרכזת ההוראה. בנוסף, האתר מהויה עborcum ערוץ תקשורת עם צוות ההוראה ועם סטודנטים אחרים בקורס. פרטיהם על למידה מותוקשבת ואתר הקורס, תמצאו באתר שה"ם בכתובת:

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

מידע על שירותים ספרייה ומקורות מידע שאהוניברסיטה מעמידה לרשותכם, תמצאו באתר הספרייה באינטרנט www.openu.ac.il/Library

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

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

لتשומת לב הסטודנטים הלומדים בחו"ל:

למרות הריחוק הפיזי הגדל, נשתדל לשמור אתכם על קשרים הדוקים ולעמוד לרשותכם ככל האפשר.

הפרטיהם החיווניים על הקורס נכללים בחוברת הקורס וכן באתר הקורס. מומלץ מאד להשתמש באתר הקורס ובכל אמצעי העזר שבו וכਮובן לפחותנו אליו במידת הצורך.

אני מאהלת לך לימוד פורה ומהנה.

ברכה,

מיכל אבימור
מרכזת ההוראה בקורס.

1. לוח זמנים ופעילותות (20465 / ב' 2021)

תאריך אחרון לשלוח ממ"ן (למנהל)	מפגשי הנקה*	יחידת הלימוד המומלצת	תאריכי שבוע הלימוד	שבוע לימוד
	מפגש ראשון	C ספר פרק 1-2-3	05.03.2021-28.02.2021	1
		C ספר פרק 1-2-3	12.03.2021-07.03.2021	2
	מפגש שני	C ספר פרק 4	19.03.2021-14.03.2021	3
ממ"ן 21.03.2021		C ספר פרק 4	26.03.2021-21.03.2021	4
	מפגש שלישי	C ספר פרק 5	02.04.2021-28.03.2021 (א-ו פסח)	5
		C ספר פרק 5	09.04.2021-04.04.2021 (ה יום הזיכרון לשואה)	6
	מפגש רביעי	C ספר פרק 6	16.04.2021-11.04.2021 (ד יום הזיכרון, ה יום העצמאות)	7
ממ"ן 18.04.2021		C ספר פרק 6	23.04.2021-18.04.2021	8
	מפגש חמישי	C ספר פרק 6,7	30.04.2021-25.04.2021 (ו ליג בעומר)	9

* התאריכים המדויקים של המפגשים הקבועתיים מופיעים בלוח מפגשים ומנהים.

לוח זמנים ופעילויות - המשך

תאריך אחרון לשלוח המסמך (למנהל)	מפגשי ההנחייה*	יחידת הלימוד המומלצת	תאריכי שבוע הלימוד	שבוע הלימוד
		C ספר 7 פרק 7	07.05.2021-02.05.2021	10
22 09.05.2021	מפגש שישי	C ספר 7 פרק 7	14.05.2021-09.05.2021	11
		C ספר + 8 פרק 8 פרויקט	21.05.2021-16.05.2021 (בשבועות)	12
23 23.05.2021	מפגש שביעי	פרויקט וחזרה	28.05.2021-23.05.2021	13
		פרויקט וחזרה	04.06.2021-30.05.2021	14
15.08.2021 **14**	מפגש שמיני	פרויקט וחזרה	11.06.2021-06.06.2021	15

מועד ב chinot הגמר יפורסם בנפרד

* התאריכים המדויקים של המפגשים הקבועתיים מופיעים בלוח מפגשים ומנהים**.

** לא תינן דחיה בהגשת הפרויקט, פרט למקורים חריגים של מילואים או מחלה ממושכת, במרקם אלו יש לתאם את מועד ההגשה עם מנהה הקבועה.

2. תיאור המטלות

על מנת לתרגל את החומר הנלמד ולבודק את ידיעותיך, عليك לפתור את המטלות המצוירות בחוברת המטלות.

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

מספר	שם	משקל
3,2,1	4 (ממ"נ חובה)	11
5,4	5 (ממ"נ חובה)	12
6,5,4	8 (ממ"נ רשות)	22
8,7,6	12 (ממ"נ רשות)	23
16	6 (ממ"נ חובה) פרויקט גמר	14

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 למנהל במפגשי ההנחיה. במקרה כזה יש לתאם את הדבר עם הבוגר).

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

הנחיות לכתיבת מטלות וניקוזן

ניקוז המטלות ייעשה לפי המשקלים הבאים :

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

ב. תיעוד - 20%

התיעוד ייכתב בתוך הקוד. אין להוסיף העורות בקבצים נפרדים.
התיעוד כולל :

- (1) הערה בראש תכנית שתכלול תיאור תמציתי של מטרות התכנית, כיצד מושגת מטרה זו, תיאור המודלים והאלגוריתם, קלט/פלט **ולכל הנחה** שהנכאים מניחים.
- (2) לכל מציג (אב-טיפוס) prototype של פונקציה (בקובץ header הצמוד לקוד), יוצמד תיאור של קלט/פלט, ופעולות הפונקציה. **מטרה :** זהו קובץ היוצר ועל כן עליו להסביר למי שאין לו גישה לקוד איך עליו להשתמש בפונקציה.
- (3) לפני הcoterrat (header) של כל פונקציה יבוא תיאור של פעולה, הנחות ואלגוריתם.

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

- 4) לכל משתנה יהיה שם שימושי ווומד אליו תיעוד לגבי תפקידו בתכנית. k,j,i - משמשים בד"כ כשמות אינדקסים ואין צורך לתרען אותם.
- 5) לא יופיעו "מספר קסם" בגין התבנית למעט 0,1 לאייחול משני ללוואות. יש להשתמש בקבועים בעלי שמות שימושיים שיכתו באותיות גדולות, ויתעדו בשלב ההגדלה. כל טיפוס מורכב יוגדר כ- `typedef` ויתעד. נהוג לקרוא לטיפוסים מורכבים בשמות שימושיים ולהשתמש באותיות גדולות.
- 6) יש להשתמש כשמות שימושיים ל: פונקציות, מקרים, משתנים, קבועים, הגדרת טיפוסים וקבצים.
- 7) יש להקפיד על קריאות ובהירות תוך שימוש באינדנטציה (היסח) מסודרת ואחדיה.

ג. תוכנות - 40%
יש להקפיד על כתיבה מסודרת ומודולרית של קוד :

- חלוקה לקבצים - ככלכל קובץ מוצמד קובץ header אם צריך (כאשר נדרש בתרגילים).
- חלוקה לפונקציות.
- שימוש במרקווים.
- שימוש נכון ב-`MAKEFILE`, (במיוחד כאשר אתם צריכים לחלק את התוכנית במספר קבצים, במסגרת הממ"ח).
- הסורת אינפורמציה - ושימוש בהפשטה מידע.
- הימנעות מכל שניינו שימוש במשתנים גלובליים.
- שימוש מירבי וכוכן במלוא הכלים שמעמידה השפה לרשותכם.
- קוד אלגנטוי ולא מסורבל.

ד. יעילות התכנית והתרשומות כללית - 20%

המקלים הנ"ל מהווים קו מנהה לחוקת הנקודות. מובן שההיה התיחסות לכל תכנית לגופה, בהתאם למידת המורכבות של התרגילים.

יתנו קנסות במיקרים הבאים :

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

لتשומת לבך : חל איסור מוחלט של הכנה משותפת של מטלות או העתקת מטלות. תלמיד שיתפס 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 בשלשות!), כאשר שני הסטודנטים המגיעים שיכים אותה 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 ובפרויקט הגמר.
- ג. ציון סופי בקורס של 60 נקודות לפחות.

لتשומת לבכם!

כדי לעודדכם להגיש לבדיקה מספר רב של מטלות הנגנו את החקלה שלהלו :

אם הגשתם מטלות מעל למשקל המינימלי הנדרש בקורס, **המטלות בציון הנמוך ביותר, שצויינה נוכחים 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 מוחשב רק לסטודנטים שעברו את בחינת הגמר והפרויקט בציון 60 ומעלה והגישו מטלות כנדרש באותו קורס.

מטלת מנהה (ממ"ז) 11

הקורס: 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטלה: פרקים 1,2,3

משקל המטלה: 4 נקודות (חוובה)

מועד אחרון להגשה: 21.03.2021

מספר השאלות: 2

סמסטר: 2021/2022

קיימות שתי חלופות להגשת מטלות:

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס

- שליחת מטלות באמצעות דואר אלקטרוני, אישור המנהה בלבד

הסבר מפורט ב"נוהל הגשת מטלות מנהה"

יש לקמפל עם דגמים מקסימליים, לקבלת כל האזהרות: `Wall-ansi-pedantic`. יש להגיש את קבצי המקור (c, h), קבצי הרצאה (את קבצי o. אין צורך לצרף), קבצי הסביבה המתאימים (כולל קבצי `makefile`), וכן קבצי קלט ותדפסי מס' או קבצי פלט (לפי הנחיות במטלה/במבחן/באטר). כל תוכנית תהיה בתיקיה נפרדת. נדרש שם התיקייה ושם הקובץ לריצה יהיו כשם הקובץ המכיל את הפונקציה `main`, ללא הסיומת `c`.

יש להגיש תוכניות מלאות (בין השאר מכילות `main`), הניתנות להידור והרצאה, ומאפשרות בדיקה של כל תוצאות הריצה המגוונות ללא צורך בשינויים כלשהם בקוד התוכנית.
את המטלה יש להגיש בקובץ `zip`. לאחר ההגשה, יש להוריד את המטלה משרת האו"פ למחשב האישי, ולבזוק שהקבצים אכן הוגשו באופן תקין.

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

עליכם כתוב תוכנית הקולעת מהקלט הסטנדרטי טקסט (רצף בקוד אסקוי), עד EOF (סוף הקלט),
ומדפיסה את הקלט לפלט הסטנדרטי בשינויים הבאים:

1. בתחילת כל משפט, אם התו הראשון הוא אות אלפביתית קטנה, יש להמיר אותה גדולות.
2. בכל טקסט בין מרווחות כפולות יש להמיר כל אות אלפביתית קטנה לאות גדולה.
3. בכל מקום אחר בטקסט (שלא לפי סעיפים 1-2 לעיל), יש להמיר כל אות אלפביתית גדולה לאות קטנה.
4. ספרות (התווים '0'-'9') לא יודפסו לפלט, ויש לדלג עליהם (לא יודפס דבר במקומן).
5. כל תו (קוד אסקוי) שאינו אות אלפביתית או ספרה יודפס ללא שינוי, לרבות סימני פיסוק וכל התווים הלבנים (רווח, טאב, שורה-חדש) בכל מקום בטקסט

דרישות נוספות:

- משפט מסתויים בתו נקודה ('.') בלבד.
- משפט חדש מתחילה בתו הראשן שאינו בן אחורי סוף המשפט הקודם.
- התו נקודה המופיע בטקסט בין מראות כפולות אינו נחשב כסוף המשפט.
- משפט יכול להתרפרש על יותר משורה אחת בקלט.
- טקסט בין מראות כפולות יכול להתרפרש על יותר משורה אחת בקלט.
- מותרונות שורות ריקות בקלט (יודפסו לפט בשורת ריקות).
- מותרים משפטיים "רייקים", המכילים מכילים רק נקודה. יש להדפיס גםמשפט רייק.
- מספר השורות בקלט בלתי מוגבל, ואורך כל שורת קלט בלתי מוגבל.
- הקלט מסתויים כשהתכנית מזזה בקלט מצב EOF במקלדת באמצעות הקלדה של צרוף המקשים `ctrl+d` באובונטו, או `ctrl+z` בחלונות.
- הקלט יכול להסתיים (EOF) גם באמצעות, למשל ללא נקודה, ואף ללא סגירת מראות.

לדוגמה, עבור הקלט הבא (9 שורות):

I am young. You are young. All of us are young.
"I think we need some help. Please" HELP. NO, NO NO,
I DO NOT
NEED HELP

WHATSOEVER.
"Today's date is
15/2/2021"....
I am 18 years old, are you 20 years old? Maybe 30 years?

יודפס הפלט הבא (9 שורות):

I am young. You are young. All of us are young.
"I THINK WE NEED SOME HELP. PLEASE" help. No, no no,
i do not
need help

whatsoever.
"TODAY'S DATE IS
//..."
I am years old, are you years old? maybe years?

על התוכנית **להציג הודעה ייחודית לקלט המפרט מה על המשמש להקליד**.
הנicho כי הקלט תקין, כלומר אין צורך לבדוק שגיאות בקלט.

הקלט לתוכנית הוא `stdin`, ויכול להציג מהמקלדת או מקובץ (באמצעות redirection בעט הפצת התוכנית). לנוחיותכם, הכינו מספר קבצי קלט והשתמשו בהם שוב ושוב לדיבוג התוכנית.

חובה **לצוף להגשה הרצות בדיקה** (אחד או יותר), המציגות את פעולת התוכנית על קלט מגוון.
יש להציג את כל השינויים 5-1 המתוירים לעיל, תוך עמידה בכל הדרישות הנוספות לעיל.
יש להציג תדפסי מסך של כל ההרצאות. אם תשימושו בקבצי קלט, יש להציג גם אותם.

שאלה 2 (תכנית ראשית בקובץ c my_sin (50 נקודות)

$$\sin(x) = \sum_{i=0}^{\infty} (-1)^i x^{2i+1} / ((2i+1)!)$$

טור טיילור לחישוב סינוס מוגדר כך:

$$x^1 / 0! - x^3 / 3! + x^5 / 5! - x^7 / 7! + x^9 / 9! \dots$$

חישוב הטור מתבצע כך:

ltahsomat lab: x natan bichidot shel radianiim (1 radian shava l- 57.296 matalot beurak).

עליכם לכתוב פונקציה (x double my_sin(double), המשמשת בטור הניל לחישוב הסינוס של x בדיק ש 0.000001, כלומר עד אשר ערכו המוחלט של האיבר הבא בטור יורד מתחת ל- 0.000001. מומלץ למש באופן יעיל, כלומר בלי לחשב מהתחלת את החזקה ואת העזרת לכל איבר בטור. אסור להיעזר בחישוב בפונקציות מהספרייה הסטנדרטית.

כמו כן, עליכם לכתוב תכנית ראשית (main), הקולעת מהמקלדת ערך ייחיד מטיפוס double וקוראת לפונקציה my_sin עם ערך זה כפרמטר. התכנית הראשית תדפיס למסך הودעת פלט נאה, הכוללת את הערך מהקלט, את תוצאת הפונקציה my_sin, וכן, לצורך ביקורת, את תוצאת הפונקציה sin המוגדרת בספרייה הסטנדרטית math.h. (הערה: לפונקציה sin אותו אב-טיפוס כמו sin_my).

אין לבצע קלט/פלט מותוך הפונקציה sin_my.

על התוכנית **להציג הודעה בקשה ייזוותית לקלט**, המפרטת מה על המשתמש להקליד. שימוש lab: כדי למנוע גישה אրיתמטית בחישוב הסינוס, יש להגביל את הערך המועבר בקלט בתחום ממשי [−25.0, 25.0]. הניחו כי הקלט תקין, כלומר אין צורך לבדוק שגיאות בקלט.

הערות טכניות נוספות:

- לביצוע קלט של ערך מטיפוס double באמצעות scanf באמצעות יש להשתמש בפורמט f %.%
- לביצוע פלט של ערך מטיפוס double באמצעות scanf באמצעות יש להשתמש בפורמט f %.
- יש לקמפל את התכנית עם הדגל lm- (בנוסח לדגמים הסטנדרטיים שצינו בפתח של הממ"ז), וזאת כדי לאפשר שימוש בפונקציה sin מ הספרייה הסטנדרטית math.h.

חוובה **לצרף להגשה מספר הרצות בדיקה**, המדגימות את פעולת התוכנית על מגוון ערכים מטיפוס double (במגבלות התחומים כמפורט לעיל). **יש להציג תדפסי מסך של כל הרצות.**

לזהירותם: לא תינטו דחיה בהגשת הממ"ז, פרט למקרים מיוחדים כגון מלאים או מחלה ממושכת. במקרים אלו יש לקבל אישור הגשה מצוות הקורס.

בהצלחה!

מטלת מנהה (ממ"ז) 12

הקורס: 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטלת: פרקים 4,5 וباופן חלקו 6

משקל המטלת: 5 נקודות (חוובה)

מספר השאלות: 1

מועד אחרון להגשה: 18.04.2021

סמסטר: 2021/2022

קיימות שתי חלופות להגשת מטלות:

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

יש לקמפל עם דגלים מקסימליים, לקבלת כל האזהרות: `-Wall -ansi -pedantic`. יש להציג את קבצי המקור (`c, h, i`), קבצי הוראה (את קבצי `s`. אין צורך לצרף), קבצי הסביבה המתאימים (כולל קבצי `makefile`), וכן קבצי קלט ותדפסי מסך או קבצי פלט (לפי הנחיות במטלה/במפגש/אתר).

כל תוכנית תהיה בתיקיה נפרדת. נדרש שם התקינה ושם הקובץ לריצה יהיו בשם הקובץ המכיל את הפונקציה `main`, ללא הסיומת `c`.

יש להגיש תכניות מלאות (בין השאר מכילות `main`), הנitinנות להידור והרצה, ומאפשרות בדיקה של כל תוצאות הריצה המגוונות ללא צורך בשינויים כלשהם בקוד התוכנית.

את המטלת יש להגיש בקובץ `zip`. לאחר ההגשה, יש להוריד את המטלת משרת האו"פ למחשב האישי, ולבודק שהקבצים אכן הוגשו באופן תקין.

שאלה 1 (תכנית ראשית בקובץ `myText.c`)

עליכם כתוב תכנית, הקוראת מהקלט הסטנדרטי טקסט המורכב מותווים (קוד ASCII). כמוות התווים בקלט **אין** **ידועה** **מראש** **ואינה** **חסומה**. על התכנית לסייע את הקריאה רק כאשר מזוהה בקלט מצב של EOF. לאחר קריאת כל הטקסט (כל התווים), על התכנית להדפיס את הטקסט לפלט הסטנדרטי, בשורות בעלות אורך קבוע של 60 תווים.

текסט נשמר בזכרון התכנית, מבנה נתונים דינמי (כלומר מבנה הנitin להגדלה לפי הצורך), כדי שיויסבר בהמשך. במקורה והקלט אורך 매우��, ואין יותר מקום בזכרון להגדיל את מבנה הנתונים, יש לעצור את הקלט, להדפיס את הטקסט שנשמר עד כה במבנה הנתונים, ולסייע את התכנית עם הודעה שגיאה נאה וברורה.

עליכם למש את מבנה הנתונים בשתי שיטות שונות. בתחילת התכנית, יש לבקש מהמשתמש לבחור באחת משתי השיטות עבור הריצה הנוכחיית. להלן תיאור השיטות.

א. חוץ (buffer) ייחד שיכיל את כל הטקסט, ויקצה באמצעות פונקציית הספרייה הסטנדרטית `calloc`. אם נגמר המקום בחוץ תוך כדי הקלט, החוץ יוגדל באמצעות פונקציית הספרייה הסטנדרטית `realloc`. הגודל ההתחלתי של החוץ יוגדר קבוע בתכנית, יהיה לפחות 60 בתים. כל הגדלה של החוץ תוסיף עוד כמה בתים בגודל ההתחלתי.

ב. רישמה מקוורת של איברים, כאשר כל איבר הוא חוץ בגודל קבוע, המכיל קטע מהtekסט. הגודל של החוצצים יוגדר קבוע בתכנית, יהיה לפחות 60 בתים. הקצתה הראשונית היא של רישמה עם איבר אחד. אם נגמר המקום באיבר (חוץ) הנוכחי תוך כדי הקלט, יוקצה איבר נוסף באמצעות הפונקציה `realloc`, ויחובר לרישמה המקוורת.

רמז : למימוש בניי הנתונים אפשר להשתמש בכלים מפרק 6 בספר הלימוד.
שימו לב : יש למש את שתי השיטות באותה התכנית, ולא בשתי תכניות נפרדות.

עליכם לכתוב את הפונקציות הבאות :

א. פונקציה בשם `readText`, הקוראת את הטקסט מהקלט הסטנדרטי, ושומרת אותו במבנה הנתונים. הפונקציה מקבלת כפרמטרים את סוג מבנה הנתונים הנבחר (אחד משני המימושים לעיל), ומצביע אל מבנה הנתונים (הकצתה ההתחלתי של מבנה הנתונים תבוצע בתכנית הראשית). הפונקציה מגדילה את מבנה הנתונים לפי הצורך תוך כדי הקלט. אם לא ניתן לקבל עוד תוספת זיכרוון באמצעות `realloc/calloc` (ראה תאור מבנה הנתונים לעיל), הפונקציה מפסיקת את הקלט, ומחזירה קוד שגיאה. הטקסט שכבר נקלט, נשאר במבנה הנתונים. הפונקציה לא מבצעת כל פלט.

ב. פונקציה בשם `printText`, שופעלת אחרי גמר הקלט, ומדפיסה לפלט הסטנדרטי את כל הטקסט שנמצא במבנה הנתונים. הפונקציה מקבלת כפרמטרים את סוג מבנה הנתונים ומצביע אל המבנה. שורות הפלט יהיו אורך אחד של 60 תווים (מלבד أول השורה האחרונות).

מהלך התכנית הראשית (`main`) הוא כדלקמן. התכנית מבקשת מהמשתמש לבחור את שיטת המימוש של מבנה הנתונים, ואז מקצתה את מבנה הנתונים הנבחר, בגודל ההתחלתי. בהמשך, התכנית מבקשת מהמשתמש להתחיל להעביר את הטקסט. ומבצעת את הפונקציה `readText` לקרוא את הטקסט מהקלט, ולאחר מכן את הפונקציה `printText` (כל פונקציה תופעל פעם אחת בלבד). לבסוף, אם הפונקציה קוד שגיאה, התכנית הראשית תדפיס הודעה שגיאה ברורה ונאה (אחרי הדפסת הטקסט).

דרישות נוספת :

- האורך של שורת קלט, ומספר השורות בקלט, אינם ידועים מראש ואיינטחסומים.
- הקלט מסתיים כשהתכנית מזהה בקלט מצב EOF. אפשר לדמות מצב EOF במקלדת באמצעות הקלדה של צרוף המקלים `ctrl+d` באובונטו, או `z+ctrl+z` בחלונות

- הטקסט בקלט יכול להכיל את כל קודי האסקוי, מלבד התו '\0' (אפשר להניח שהקלט תקין).
- כל התווים מהקלט יישמרו במבנה הנטוניים, מלבד התו '\n' (שורה-חדרה), עליו יש לדלג בכל פעם שהוא מופיע, ולא להכניסו לבנייה.

אפשר ומומלץ (לא חובה) למש פונקציות נוספות. למשל, פונקציות עוזר להגדלת מבנה הנטוניים בכל אחת מהשיטות, ועוד'.

על התוכנית **להציג הודעת בקשה יידוטית בכל פעם שנדרש קלט**. כמותוар לעיל, הקלט הראשון הוא בחירת סוג המבנה, והקלט השני הוא הטקסט (מספריקה בקשה אחת לכל הטקסט).

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

חובה **לצף להגשה הרצות דוגמה** (אחד או יותר), המדגימות את פעולה התוכנית. בפרט, יש להציג טקסט ארוך בקלט, שדורש הגדלה של מבנה הנטוניים תוך כדי הקלט. **יש להציג תדפסי מסך או קבצי פלט של כל הרצות. אם תשתמשו בקבצי קלט, יש להגיש גם אותם.**

להזכירם: לא תינטו דחיה בהגשת הממיין, פרט למקומות מיוחדים כגון מילואים או מילה ממושכת. במקרים אלו יש לקבל אישור הגשה מהמנחה.

בהצלחה!

מטלת מנהה (ממ'ו) 22

הקורס: 20465 - מעבדה בתכניות מערכות

חומר הלימוד למטרת: פרקים 4,5,6

משקל המטלה: 8 נקודות (רשوت)

מספר השאלות: 1

מועד אחרון להגשתה: 09.05.2021

סמסטר: 2020/2021

קיימות שתי חלופות להגשת מטלות:

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס
- שליחת מטלות באמצעות דואר אלקטרוני, באישור המנהה בלבד

הסבר מפורט ב"נווה הגשת מטלות מנהה"

יש לкопל עם דגמים מקסימליים, לקבלת כל האזהרות: `-Wall -ansi -pedantic`. יש להגיש את קבצי המקור (c, h), קבצי הרצאה (את קבצי o. אין צורך לזרף), קבצי הסביבה המתאימים (כולל קבצי `makefile`), וכן קבצי קלט ותדפסי מסך או קבצי פלט (לפי הנחיות במטריה/במגש/באתר). כל תוכנית תהיה בתיקיה נפרדת. נדרש שם התיקייה ושם הקובץ לריצה יהיו בשם הקובץ המכיל את הפונקציה `main`, ללא הסיומת `.c`.

יש להגיש תוכניות מלאות (בין השאר מכילות `main`), הנitinנות להידור והרצאה, ומאפשרות בדיקה של כל תוצאות הריצה המגוונות ללא צורך בשינויים כלשהם בקוד התוכנית. את המטלה יש להגיש בקובץ `zip`. לאחר ההגשתה יש להוריד את המטלה משרת האו"פ למחשב האישי, ולבדוק שהקבצים אכן הוגשו באופן תקין.

שאלה 1 (תכנית ראשית בקובץ `myset.c`, ובנוסף הקבצים `set.h`, `set.c`)

עליכם לכתוב תוכנית שפועלת כ"מחשב כיס" אינטראקטיבי לביצוע פעולות על קבוצות.

תזכורת: קבוצה (`set`) היא אוסף של איברים, בני מניה, ולא חזרות (כלומר, לכל איבר בקבוצה יש ערך שונה מכל האיברים האחרים).

משימות התוכנית:

עליכם לכתוב תוכנית מחשב הקוראת פקודות מהקלט הסטנדרטי, מפענחת ומבצעת אותן. הפקודות עוסקות בפעולות על קבוצות.

עליכם להגדיר, תוך שימוש ב-`typedef`, את הтипוס `set`, אשר מחזיק קבוצה של מספרים שלמים מהנתונים הסגור [0...127]. על הтипוס (מבנה הנתונים) `set` להיות חסכוני מבחרינת כמות הזיכרון הנדרשת. כך למשל, מערך של 128 בתים אינו חסכוני.

רמז: אפשר להסתפק בסביבה (`tio`) אחת לכל איבר בקבוצה.

בנוסף עליהם להגדיר, בתכנית הראשית, שישה משתנים מטיפוס `set`, בשמות הבאים:
.SETA, SETB, SETC, SETD, SETE, SETF
בתחילת ריצת התכנית, יש לאותחל כל אחד מהםותים לקבוצה הריקה.

כעת, עליהם לבצע פעולות מגוונות על קבוצות. כל פעולה תופעל באמצעות פקודה שמועברת מהמשתמש בקלט לתכנית.

בפוקודות שיפורטו להלן, אופרנד שהוא שם של קבוצה, יהיה אחד מששת המשתנים שהוגדרו לעיל.

מבנה הפוקודות המשמשות בקלט לתכנית:

لتשומת לב: סדר קריאת השדות בפוקודה הוא משמאלי לימין.

1. הצבת איברים בקבוצה

רישימת-ערכים-מופרדים-זה-זה-בפסיקים, שם-קבוצה read_set

הפוקודה מכניסה את הערכים שברשימה לתוך הקבוצה ששם ניתן בפוקודה. רשימת הערכים אינה סדרה, ומותר לערך להופיע בה יותר מפעם אחת (מופעים חוזרים לא יילקחו בחשבון). סוף רשימת הערכים מסומן על ידי המספר השלילי -1.

לדוגמה, הפוקודה: `read_set SETA, 5, 6, 5, 76, 44, 23, 23, 98, 23, -1`

מציב במשתנה SETA את הקבוצה:

لتשומת לב: ההצבה יוצרת קבוצה **חדשה**, שמחילפה את התוכן המקורי של המשתנה.
אם הרשימה אינה מכילה אף ערך (מלבד -1 המסיים), הקבוצה שתיווצר היא ריקה.

2. הדפסת קבוצה

שם-קבוצה print_set

הפוקודה מדפסת את איברי הקבוצה ששם ניתן בפוקודה, בסדר עולה של הערכים, ולכל הייותר 16 ערכים בכל שורת פלט. יש להקפיד על פורמט נאה של הדפסה.
אם הקבוצה ריקה, יש להדפיס "The set is empty"

3. איחוד של קבוצות

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

הפוקודה מבצעת איחוד של קבוצה א' עם קבוצה ב', ואת התוצאה מאחסנת בקבוצה ג'.
תזכורת: תוצאה האיחוד היא קבוצת כל האיברים הנמצאים בקבוצה א' ו/או בקבוצה ב'.

4. חיתוך של קבוצות

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

הפוקודה מבצעת חיתוך של קבוצה א' עם קבוצה ב' ואת התוצאה מאחסנת בקבוצה ג'.
תזכורת: תוצאה החיתוך היא קבוצת כל האיברים הנמצאים גם בקבוצה א' וגם בקבוצה ב'.

5. חיסור של קבוצות

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

הפוקודה מבצעת חיסור של קבוצה ב' מקבוצה א', ואת התוצאה מאחסנת בקבוצה ג'.
תזכורת: תוצאה החיסור היא קבוצת כל האיברים הנמצאים בקבוצה א' ולא נמצאים בקבוצה ב'.

6. הפרש סימטרי של קבוצות

שם-קבוצה-א', שם-קבוצה-ב', שם-קבוצה-א' symdiff_set

הפקודה מחשבת הפרש סימטרי של קבוצה א' וקבוצה ב', ואת התוצאה מאחסנת בקבוצה ג'.
תזכורת: הפרש סימטרי הוא קבוצת כל האיברים הנמצאים בקבוצה א' או בקבוצה ב', אבל לא נמצאים בחיתוך של קבוצה א' עם קבוצה ב'.

7. עיצירת הוכןנות

stop

זהי פקודה ללא אופרנדים, הגורמת לסיום מיידי של הוכןנות.

لتשומת לב: אותו שם קבוצה יכול לשמש ביותר מאפרנד אחד באוטה הפקודה. מימוש הפעולות על קבוצות צריך להתחשב באפשרות זו (לא לדروس נתונים תוך כדי חישוב).
לדוגמא, הפעולות שלහן תקיןות ומוגדרות היטב:

union_set SETC, SETD, SETD

intersect_set SETA, SETF, SETA

sub_set SETC, SETC, SETC (מה האפקט של פעולה זו?)

union_set SETA, SETA, SETF (מה האפקט של פעולה זו?)

המבנה התחבירי של הקלט:

- כל פקודה תופיע בשורתה בשורת קלט ייחידה, כולל כל האופרנדים. מותרות גם שורות ריקות (שורות המכילות רק טווים לבנים).
- שם הפקודה מופרד מהאופרנד הראשון באמצעות רווחים ו/או טאים (אחד או יותר).
- בין כל שני אופרנדים של הפעולה יש פסיק אחד. לפני ואחרי הפסיק יכולים להיות רווחים ו/או טאים בכמות בלתי מוגבלת. אסור שהייה פסיק אחרי האופרנד האחרון.
- יכולים להיות רווחים ו/או טאים בכמות בלתי מוגבלת בתחילת השורה (לפני שם הפקודה), וגם בסוף השורה (אחרי האופרנד האחרון).
- אסור שהיו טווים מיותרים (תווי זבל) בסוף השורה, למעט טווים לבנים.
- שמות הפקודות יופיעו באותיות קטנות בלבד, ושמות הקבוצות באותיות גדולות בלבד.

אופן פעולות הוכןנות:

יש למשך ממשק משתמש ידידותי, כך שהמשתמש יוכל להבין בכל שלב של הוכןנות מה עליו לעשות. בפרט, על הוכןנית להדפיס הודעה או סימן (prompt) בכל פעם שהיא מוכנה לקבל את הפקודה הבאה. הוכןנית תמשיך לקלוט ולבצע פקודה אחריה פקודה, עד שתתקבל הפקודה stop.

הוכןנית אינה מניחה שהקלט תקין. על הוכןנית לנתח כל פקודה ולזוזה שאין בה שגיאות (ראו דוגמאות בהמשך). במידה ונתקלה שגיאה, הוכןנית תדפיס הודעה שגיאה פרטנית, ותעביר לפקודה הבאה, בלי לבצע את הפקודה השגויה. אין לעצור את הוכןנית עם גילוי השגיאה הראשונה. אין צורך על יותר משגיאה אחת בכל שורת קלט.

יש לטפל גם במצב של EOF (גמר הקלט). עיצירת הוכןנית שלא באמצעות stop מפזרת בקלט אינה נחשבת תקינה (גם לא כאשר הקלט מגע מקובץ באמצעות redirection), ויש להדפיס הודעה שגיאה על כך ורק אז לעצור.

שימוש לב: השורה الأخيرة בקובץ קלט אינה חייבת להסתיים בתו 'ת'.

להלן דוגמאות של קלט שגוי:

שימוש לב: יתכנו סוגים נוספים של שגיאות בקלט.
עליכם לחושב על כל מגוון השגיאות האפשריות, ולטפל בכלן.

1. **לפקודה:**
read_set SETG, 3, 6, 5, 4, 4, -1
יש להגיב בהודעה כגון:
Undefined set name
2. **לפקודה:**
read_set setA, 3, 6, 5, 4, 4, -1
יש להגיב בהודעה כגון:
Undefined set name
3. **לפקודה:**
do_it SETA, SETB, SETC
יש להגיב בהודעה כגון:
Undefined command name
4. **לפקודה:**
UNION_set SETA, SETB, SETC
יש להגיב בהודעה כגון:
Undefined command name
5. **לפקודה:**
read_set SETB, 45, 567, 34, -1
יש להגיב בהודעה כגון:
Invalid set member – value out of range
6. **לפקודה:**
read_set SETA, 45, 56, 45, 34
יש להגיב בהודעה כגון:
List of set members is not terminated correctly
7. **לפקודה:**
read_set SETA, 45,-3, 2, 45, 34, -1
יש להגיב בהודעה כגון:
Invalid set member – value out of range
8. **לפקודה:**
read_set SETA, 45, 2, xyz, 34, -1
יש להגיב בהודעה כגון:
Invalid set member – not an integer
9. **לפקודה:**
read_set SETA, 45, 2, 24.0, 34, -1
יש להגיב בהודעה כגון:
Invalid set member – not an integer
10. **לפקודה:**
union_set SETC, SETA
יש להגיב בהודעה כגון:
Missing parameter
11. **לפקודה:**
union_set SETC, SETA, SETB,
יש להגיב בהודעה כגון:
Extraneous text after end of command
12. **לפקודה:**
print_set SETC, SETD
יש להגיב בהודעה כגון:

Extraneous text after end of command

13. לפקודה :

sub_set SETF, , SETD, SETA

יש להגיב בהודעה כגון :

Multiple consecutive commas

14. לפקודה :

intersect_set SETF SETD SETA

יש להגיב בהודעה כגון :

Missing comma

15. לפקודה :

symdiff_set, SETF, SETB, SETA

יש להגיב בהודעה כגון :

Illegal comma

להלן דוגמה של סדרת פקודות שכולו תקיןות :

הערה : סידרה כגון זו יכולה לשמש כקלט להרצת בדיקה של נכונות הביצוע של הפקודות (ללא טיפול בשגיאות בקלט).

```
print_set SETA
print_set SETB
print_set SETC
print_set SETD
print_set SETE
print_set SETF
read_set SETA, 45, 23, 6, 7, 4, 3, 75 ,45, 34, -1
print_set SETA
read_set SETB, 5, 4, 3, 2, 78, 45, 43, -1
print_set SETB
read_set SETC,100,105,101,103,104,-1
print_set SETC
read_set SETC,127,0,3,126,127,0,-1
print_set SETC
read_set SETC,-1
print_set SETC
read_set SETD, -1
print_set SETD
read_set SETC , 110 , 111, 112 , -1
print_set SETC
union_set SETA, SETB, SETD
print_set SETD
intersect_set SETA, SETB, SETE
print_set SETE
sub_set SETA, SETB, SETF
print_set SETF
symdiff_set SETA, SETB, SETF
print_set SETF
intersect_set SETA, SETC, SETD
```

```

print_set SETD
union_set SETB, SETB, SETE
print_set SETE
intersect_set SETB, SETA, SETB
print_set SETB
union_set SETA, SETC, SETC
print_set SETC
symdiff_set SETC, SETA, SETC
print_set SETC
sub_set SETC,SETC,SETC
print_set SETC
union_set SETF , SETC , SETF
print_set SETF
stop

```

ארגון קוד התוכנית:

יש לחלק את התוכנית למספר קבצי מקור : .set.h → , set.c , myset.c

- בקובץ set.c יש לרכז את הפעולות על קבוצות. לכל פעולה יש לממש פונקציה נפרדת, ששם הפקודה בקלט (לדוגמה : print_set ,read_set ,union_set ,etc). האופרנדים של הפעולה יועברו כפרמטרים של הפונקציה, לפי מפרט הפעולה כמוגדר לעיל. לא יבוצע כל קלט/פלט בקובץ זה, ולא ניתוח תחבירי של הקלט.
- בקובץ myset.c יהיה הפונקציה main , וכן כל פעילות האינטראקטיבית עם המשתמש, ניתוח הפקודות, והדפסת הודעות השגיאה. כמו כן, בקובץ זה יוגדרו שששת המשתנים מטיפוס set .
- בקובץ set.h תהיה הגדרת טיפוס הנתונים set , וכן הCHARACTERS (אב טיפוס) של הפונקציות המומושות בקובץ set.c . יש לכלול (#include) את הקובץ set.h בקבצי המקור האחרים.
- אפשר לבנות קבצי מקור נוספים (למשל : קובץ המכיל פונקציות עוזר לניתוח הקלט, וכו'). הקלט לתוכנית הוא `stdin`, ויכול להציג מהמ Kendall או מקובץ redirection (באמצעות redirection בעט הפעלת התוכנית). לנוחיותכם, הינו מספק קבצי קלט והשתמשו בהם שוב ושוב לדיבוג התוכנית. בכל קובץ קלט תהיה סדרה של פקודות על קבוצות.
- על התוכנית להזעט בקשה יידידותית לקלט עבור כל שורת קלט (כל פקודה). כמו כן, לפני הניתוח של שורת הקלט, על התוכנית להציג את השורה לפט בזיהוק כפי שנקרה. זאת כדי שנייתן יהיה לראות בפלט את הפקודות המקוריות, גם כאשר הקלט מגע מקובץ.
- חובה לצרף להגשה הרצאות בדיקה (אחד או יותר), המדגימות את השימוש בכל הפעולות על קבוצות ובכל ששת הקבוצות המוגדרות, וכן את הטיפול בכל מגוון השגיאות בקלט. רמז : מומלץ להכניס בקלט פקודת הדפסה של קבוצת התוצאה אחורי כל פעולה, כדי להראות שההתוצאה אכן נכונה (ראו לעיל הדוגמה של סדרת פקודות תקיןות).
- יש להציג תדפיס מסך (או קובץ פלט) של כל הרצאות. אם השתמשו בקבצי קלט, יש להגיש גם קבצים אלה.

להזכירכם : לא ניתן דחיה בהגשת המ"ן, פרט לקרים מיוחדים כגון מלאים או מחלה ממושכת. במקרים אלו יש לקבל אישור הגשה מהמנהה.

בהצלחה!

מטלת מנהה (ממ'ו) 23

הקורס : 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטלה : פרקים 6,7,8

משקל המטלה : 12 נקודות (רשوت)

מספר השאלות : 2

מועד אחרון להגשה : 23.05.2021

סמסטר : 2021/2022'

קיימות שתי חלופות להגשת מטלות :

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס
- שליחת מטלות באמצעות דואר אלקטרוני - **באישור המנהה בלבד**

הסבר מפורט ב"נווה הגשת מטלות מנהה"

יש לקמפל עם דגמים מקסימליים, לקבלת כל האזהרות: `-Wall -ansi -pedantic`. יש להגיש את קבצי המקור (c, h, h.), קבצי הרצה (את קבצי o. אין צורך לצרף), קבצי הסביבה המתאימים (כולל קבצי `makefile`), וכן קבצי קלט ותדפסי מסך או קבצי פلت (לפי הנחיות במטלה/במفوض/ באתר). כל תוכנית תהיה בתיקיה נפרדת. נדרש שם התיקייה ושם הקובץ לריצה יהיו כשם הקובץ המכיל את הפונקציה `main`, ללא הסיומת `.c`.

יש להגיש תוכניות מלאות (בין השאר מבילות `main`), ניתנות להידור והרצה, ומאפשרות בדיקה של כל התוצאות המגוונות של הריצה ללא צורך בשינויים כלשהם בקוד התוכנית.
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 عليיכם כתוב האם הטעונה נכון, לא נכונה, לפערם נכון. עליכם לנמק את תשובהכם.

תשובה לא מנומקת, גם אם היא נכון, לא תזכה בנקודות. (כל סעיף 5 נקודות).

א. הביתוי

`#define STR1 'a'`

שколо למשה לביטוי :

`#define STR1 "a"`

ב. פונקציה יכולה לשנות את ערכי הפרמטרים המועברים אליה בקריאה, והשינויים ייראו מחוץ לפונקציה אחרי החזרה מהקריאה.

את הפתרון לשאלה זו יש להגיש במסמך (קובץ) מוקלד, בכל פורמט.

שאלה 2

(תבנית ראשית בקובץ prnt.c) (90 נקודות)

תזכורת : כאשר מהדרים (compile) תוכנית בשפת C הבנייה מקובץ מקור יחיד, מקובל ליצור קובץ ביצוע (executable) עם שם זהה לקובץ המקור, אך עם סיממת שונה, או ללא סיממת בכלל.

לדוגמה : כתבו תכנית ושמרו אותה בקובץ בשם prnt.c, והידרנו אותה (לא שגיאות) באמצעות הפקודה :
gcc -ansi -pedantic -Wall prnt.c -o prnt
ונוצר קובץ ביצוע בשם prnt (באובונטו) או prnt.exe (בחלונות).

עליכם כתוב תכנית בקובץ מקור יחיד, המדפיסה את קובץ המקור של עצמה.
למשל, בדוגמה לעיל, הרצת התוכנית ותדפיס לפלט הסטנדרטי את תוכן הקובץ prnt.c.

הערות :

1. הניחו שני הקבצים, קובץ המקור וקובץ הביצוע, שמורים באותה תיקיה.
2. על התוכנית לעבוד נכון לא צורך להעביר אליה ארגומנטים בשורת הפקודה, וגם ללא redirection של הקלט הסטנדרטי בעת הפעלת התוכנית.
3. ייתכן ובעתיד תמצאו לנכון להחליף את שמות הקבצים של התוכנית. בהנחה, שהחלפת השמות נעשית באופן מתואם, כלומר קובץ המקור וקובץ הביצוע מקבלים את אותו שם חדש (למעט הסיממת), על התוכנית שלכם להמשיך לעבוד נכון, ללא צורך בשינויים כלשהם בקובץ המקורי, ולא צורך בהידור מחדש.

חובה לצרף להגשה הרצת בדיקה, המדגימה את פועלות התוכנית. יש להגיש תדפיס מסך מלא (או קובץ פלט) של ההרצה.

להזיכרים : לא תינטו דחיה בהגשת הממיין, פרט למקרים מיוחדים כגון מילואים או מחלת ממושכת.
במקרים אלו יש לקבל אישור הגשה מצוות הקורס.

בהצלחה!

מטלת מנהה (ממ"ו) 14

הקורס : 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטרלה : פרויקט גמר

משקל המטרלה : 61 נקודות (חובה)

מספר השאלות : 1

מועד אחרון להגשה : 15.08.2021

סמסטר : ס' 2021/2022

קיימות שתי חלופות להגשת מטלות:

- שליחת מטלות באמצעות מרכיבת המטלות המקוונת באתר הבית של הקורס
- שליחת מטלות באמצעות דואר אלקטרוני - באישור המנהה בלבד

הסביר מפורט ב"נוהל הגשת מטלות מנהה"

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

עליכם לכתוב תוכנת אסמבילר, עברו שפת אסמבלי שתוגדר בהמשך. הפרויקט ייכתב בשפת C.

עליכם להגיש את הפריטים הבאים :

1. קבצי המקור של התוכנית שכתבתם (קבצים בעלי סיומת .c או .h).
2. קובץ הרצה (מוקומפל ומקשור) עברו מערכת אוביונטו.
3. קובץ makefile. הקימפול חייב להיות עם הקומפיילר gcc והdaglim : -Wall -ansi -pedantic . יש לנפות את כל ההודעות שמוציאה הקומפיילר, כך שהתוכנית תתקmpl ללא כל העורות או אזהרות.
4. דוגמאות הרצה (קלט ופלט) :
 - א. קובצי קלט בשפת אסמבלי, קובצי הפלט שנוצרו מהפעלת האסמבילר על קבצי קלט אלה. יש להציגו במגוון הפעולות וטיפוסי הנתונים של שפת האסמבלי.
 - ב. קובצי קלט בשפת אסמבלי המדגימים מגוון רחב של סוגים שונים אסמבלי (ולכן לא נוצרים קבצי פלט), ותדפיסי המיצן המראים את הדוגמאות השגיאה שמוציאה האסמבילר.

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

נזכיר מספר היבטים חשובים של כתיבת קוד טוב :

1. הפשטה של מבני הנתונים : רצוי (כל האפשר) להפריד בין הגישה למבנה הנתונים לבין הIMPLEMENTATION של מבני הנתונים. כך, למשל, בעת כתיבת פונקציות לטיפול בטבלה, אין זה מעניינים של משתמשים בפונקציות אלה, האם הטבלה ממומשת באמצעות מערך או באמצעות רשימה מקוורת.
2. קריאות הקוד : יש להשתמש בשמות שימושיים למשתנים ופונקציות. יש לעורך את הקוד באופן מסודר : הזוחות עקביות, שורות ריקות להפרדה בין קטעי קוד, וכו'.

3. תיעוד: יש להכניס בקבצי המקור תיעוד תמציתי וברור, שיסביר את תפקידיה של כל פונקציה (באמצעות העורות כוורת לכל פונקציה). כמו כן יש להסביר את תפקידם של משתנים חשובים. כמו כן, יש להכניס העורות ברמת פירוט גבוהה בכל הקוד.

הערה: תוכנית "עובדת", דהיינו תוכנית שביצעת את כל הדריש ממנה, אינה לכשעצמה עрова, לזמן גבוי. כדי לקבל ציון גבוה, על התוכנית לעמוד בקריטריונים של כתיבה ותיעוד ברמה גבוהה, כאמור לעיל, אשר משקל המשותף מגע עד לכ- 40% ממשקל הפרויקט.

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

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

מומלץ לקרוא את הגדרת הפרויקט פעם ראשונה ברצף, לקבלת תמונה כללית לגבי הנדרש, ורק לאחר מכן לקרוא שוב בצורה מעמיקה יותר.

רקע כללי ומטרת הפרויקט

כידוע, קיימות שפות תוכנות רבות, ומספר גדול של תוכניות, הכתובות בשפות שונות,עשויות לזרז 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 העבודה המרכזית - היע"מ (CPU) - יודעת לפרק את הרצף הזה לקטעים קטנים בעלי משמעות: הוראות, מענים ונתונים.

למעשה, זיכרונו המחשב כולל הוא אוסף של סיביות, שנוהגים לראותם כמקובצות ליחידות בעלות אורך קבוע (למשל בית - byte). לא ניתן להבחין, בעין שайינה מיומנת, בהבדל פיזי כלשהו בין אותן חלק בזיכרון שבו נמצאת תוכנית לבין שאר הזיכרון.

יחידת העבודה המרכזית (היע"מ) יכולה לבצע מגוון פעולות פשוטות, הנקראות **הוראות מכונה**, ולשם כך היא משתמשת באוגרים (registers) הקיימים בתחום היע"מ, ובזיכרון המחשב.

דוגמאות: העברת מספר מתא בזיכרון לאוגר ביע"מ או בחזרה, הוספה 1 למספר הנמצא באוגר, בדיקה האם מספר המאוחסן באוגר שווה לאפס, חיבור וחיסור בין שני אוגרים, ועוד. הוראות המכונה ושילובים שלهنן הן המרכיבות תוכנית כפי שהיא טעונה לזכרו בזמן ריצתה. כל תוכנית מקור (התוכנית כפי שנכתבה בידי המתכנת), תתרגם בסופו של דבר באמצעות תוכנה מיוחדת לזרה סופית זו.

היע"מ יודע לבצע קוד שנמצא בפורמט של **שפת מכונה**. זהו רצף של ביטים, המהווים קידוד ביניاري של סדרת הוראות המכונה המרכיבות את התוכנית. קוד כזה אינו קרייא למשתמש, ולכן לא נוח לקודד (או לקרוא) תוכניות ישירות בשפת מכונה. **שפת אסמבלי** (assembly language) היא שפת תוכנות מאפשרת לייצג את הוראות המכונה בזרה סימבולית קלה ונוחה יותר לשימוש. מבן שיש צורך לתרגם את הייצוג הסימבולית לקוד בשפת מכונה, כדי שהתוכנית תוכל לרוץ במחשב. תרגום זה נעשה באמצעות כלי שנקרא **אסambilר** (assembler).

כידוע, לכל שפת תוכנות עילית יש מהדר (compiler), או מפרש (interpreter), המתרגמים תוכניות מקור לשפת מכונה. האסambilר משמש בתפקיד דומה עבור שפת אסambilי.

לכל מודל של ייע"מ (כלומר לכל אירוגון של מחשב) יש שפת מכונה יעודית משלו, ובהתאם גם שפת אסםביי יעודית משלו. לפיכך, גם האסםבלר (כלי התרגומים) הוא יודי ו שונה לכל ייע"מ.

תפקידו של האסםבלר הוא לבנות קובץ המכיל קוד מכונה, מקובץ נתנו של תוכנית הכתובה בשפת אסםבלאי. זהו השלב הראשון בمسلسل אותו עברת התוכנית, עד לקבלת קוד המוכן לריצה על חומרת המחשב. השלבים הבאים הם קישור (loading) וטעינה (linkage), אך בהם לא נעסק במאין זהה.

המשימה בפרויקט זה היא ל כתוב אסםבלר (כלומר תוכנית המתרגם לשפת מכונה), עבור שפת אסםבלאי שנגידר כאן במיוחד לצורך הפרויקט.

لتשומת לב : בהסבירים הכלליים על אופן העבודה תוכנת האסםבלר, תהיה מדי פעם התיאיחסות גם לעבודת שלבי הקישור והטעינה. התיאיחסות אלה נועדו על מנת לאפשר לכם להבין את המשך תהליך העיבוד של הפלט של תוכנת האסםבלר. אין לטעות: עליים ל כתוב את תוכנית האסםבלר בלבד. אין ל כתוב את תוכניות הקישור והטעינה!!!

המחשב הדמיוני ושפת האסםבלאי

נגידר עתה את שפת האסםבלאי ואת מודל המחשב הדמיוני, עבור פרויקט זה.

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

"חומרה":

המחשב בפרויקט מרכיב **מעבד** (יע"מ), **אוגרים** (רגיסטרים), **זיכרון RAM**. חלק מהזיכרון משמש כמחסנית (stack).

למעבד 32 אוגרים כלליים, בשמות: \$0,\$1,\$2....\$31. גודלו של כל אוגר הוא 32 סיביות. הסיבית הכי פחות משמעותית תצוין כסיבית מס' 0, והסיבית המשמעותית ביותר כמס' 31.

גודל הזיכרון הוא 2^{25} תאים, בכתובות 1-0, וכל תא הוא בגודל של 8 סיביות (Byte). כתובות בזיכרון ניתנת לייצוג ב-25 סיביות (מספר ללא סימן).

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

הוראות המכונה:

למחשב זה מגוון הוראות, כל הוראה מורכבת מפעולה ואופרנדים. מספר האופרנדים תלוי בסוג ההוראה.

כל הוראה מקודדת בקוד המכונה ל-32 סיביות. הסיבית הפחות משמעותית של קידוד ההוראה תצוין כסיבית מס' 0, והסיבית המשמעותית ביותר כמס' 31.

הוראה תופסת בזכרון המחשב ארבעה בתים רצופים, ומוחסנת בשיטת little-endian. ככלומר, סיביות 7-0 של ההוראה נמצאות בבית שכתוותו הנמוכה ביותר, סיביות 15-8 הן הבית הבא, לאחר מכן סיביות 23-16, ואילו סיביות 31-24 נמצאות בכתובת הגבוהה ביותר. כשמתייחסים לכתובת של הוראה בזכרון, זו תמיד כתובת הבית הנמוך ביותר.

לכל הוראה יש קוד-פעולה, הנקרא גם opcode, שמצויה את הפעולה שבוצעת ההוראה. לחלק מההוראות יש קוד זיהוי שני, שנקרא funct.

ההוראות נחלקות ל-3 סוגים : הוראות מסוג R, הוראות מסוג I, והוראות מסוג J.
להלן טבלה המפרטת את כל ההוראות, לפי סוגן, עם הקודים המזוהים שלهن :

שם הפעולה	סוג הפעולה	funcet (בסיס עשרוני)	opcode (בסיס עשרוני)
add	R	1	0
sub	R	2	0
and	R	3	0
or	R	4	0
nor	R	5	0
move	R	1	1
mvhi	R	2	1
mvlo	R	3	1
addi	I		10
subi	I		11
andi	I		12
ori	I		13
nori	I		14
bne	I		15
beq	I		16
blt	I		17
bgt	I		18
lb	I		19
sb	I		20
lw	I		21
sw	I		22
lh	I		23
sh	I		24
jmp	J		30
la	J		31
call	J		32
stop	J		63

הערה : בתחביר הוראה בשפת אסמבלי, שם-הפעולה נכתב תמיד באותיות קטנות.

מבנה הוראות המכונה :

נדון עתה ביתר פרוט במבנה התחבירי של כל הוראה בשפת האסמבלי, ובאופן הקידוד של הוראה בקוד המכונה.

הוראות מסוג R:

הוראה מסוג R מוקדדת באופן הבא :

opcode	rs	rt	rd	funct	לא בשימוש
31	26	25	20	16	11

בhorאות מסוג R כל האופרנדים הם אוגרים. חלק מההוראות יש שני אופרנדים, ולחלקם שלשה אופרנדים.

קובוצת horאות מסוג R כוללת את horאות הבאות :

- horאות אריתמטיות ולוגיות מסוג R : add, sub, and, or, nor
- horאות העתקה : move, mvhi, mvlo

הוראות אריתמטיות ולוגיות מסוג R

להוראות add, sub, and, or, nor יש שלשה אופרנדים, ושלשות אוגרים. horאות מבצעות פעולה אריתמטית או לוגית בין שני האוגרים rs ו- rt, והווצאה מאחסנת באוגר השלישי rd. לכל horאות האריתמטיות/לוגיות מסוג R יש קוד-פעולה זהה והוא 0 (כלומר שדה opcode מכיל 0). על מנת להבחין בין horאות השונות, קיימים שדה נוסף בשם funct שהוא ייחודי לכל horאה (ראו טבלת horאות לעיל).

- הוראה add מבצעת פעולה חיבור, ככלומר $rd = rs + rt$
- הוראה sub מבצעת פעולה חיסור, ככלומר $rd = rs - rt$
- הוראה and מבצעת פעולה and בין הסיביות של האופרנדים, ככלומר $rd = rs \& rt$
- הוראה or מבצעת פעולה or בין הסיביות של האופרנדים, ככלומר $rd = rs | rt$
- הוראה xor מבצעת פעולה xor בין הסיביות של האופרנדים, ככלומר $rd = \sim(rs | rt)$

נדגים את השימוש בהוראות, ואת אופן קידודן :
add \$3, \$19, \$8

הוראה זו מבצעת חיבור של תוכן האוגר 3 עם תוכן האוגר 19 ושומרת את התוצאה באוגר 8.

בקידוד horאה זו, שדה opcode יכול את קוד הפעולה 0 (המשותף לכל horאות מסוג R) ובשדה funct את הקוד הייחודי ל-add. שדה rt יכול את מספרו של האוגר המשמש כמחובר הראשון (בדוגמה זו 3), שדה rs יכול את מספרו של האוגר המשמש כמחובר השני (בדוגמה זו 19), ושדה rd יכול את מספרו של אוגר התוצאה (בדוגמה זו 8). הסיביות 5-0 בקידוד horאה אינן בשימוש, וכיילו אפסים.

באופן דומה ל-add, משתמשים גם בהוראות : xor, sub, and, or, nor .
נדגיש שוב שההוראה sub האוגר rs הוא המחוסר, והאוגר rt הוא המחסר.
הערה : אין מניעה שאותו אוגר ישמש ביותר מ一封ר אחד (למשל : or \$1,\$2,\$1).

הוראות העתקה :

להוראות move, mvhi, mvlo יש שני אופרנדים, ושניהם אוגרים. horאות מעתיקות (MSCFLOT) תוכן של אוגר אחד (אוגר המקור) לאוגר שני (אוגר היעד).

לכל ההוראות הפעתקה מסווג R יש קוד-פעולה זהה והוא 1 (כלומר שדה opcode המכיל 1). על מנת להבחין בין ההוראות השונות, קיים שדה funct ייחודי לכל אחת מהן (ראו טבלת ההוראות לעיל).

- הוראה move מעתיקה את תוכנו של אוגר rs אל אוגר rd
- הוראה mvhi מעתיקה את חציו הגבוה (סיביות 31-16) של אוגר rs אל חציו הנמוך של אוגר rd
- הוראה mvlo מעתיקה את חציו הנמוך (סיביות 15-0) של אוגר rs אל חציו הגבוה של אוגר rd

נדגים את השימוש בהוראות, ואת אופן קידודן:

move \$23, \$2

הוראה זו מעתיקה אל האוגר \$23 את תוכנו של אוגר \$2.

בקידוד הוראה זו, שדה opcode יכול את קוד הפעולה 0 (המשותף לכל ההוראות מסווג R), ושדה funct את הקוד הייחודי ל-move. השדה rs יכול את מספרו של אוגר המקור של הפעתקה (בדוגמה זו 2), והשדה rd יכול את מספרו של אוגר היעד (בדוגמה זו 23). השדה rt, וכן הסיביות 0-5 אינם בשימוש ויכילו אפסים.

. mvhi, mvlo משמשים גם בהוראות move.

הוראות מסווג I:

הוראות מסווג I מקודדת באופן הבא:

opcode	rs	rt	immed
31 26 25	21 20	16 15	0

בהוראות מסווג I אחד האופרנדים הוא ערך קבוע, הנקרא גם ערך מיידי (immediate). הקבוע הוא מספרשלם, הנitinן לייצוג ברוחב 16 סיביות בשיטת המשלים ל-2. סיבית 0 של קידוד הוראה היא הסיבית הפחות משמעותית של הקבוע.

לכל אחת מההוראות מסווג I יש ייחודי (ראו טבלת ההוראות לעיל).

קבוצת ההוראות מסווג I כוללת את ההוראות הבאות:

- הוראות אריתמטיות ולוגיות: addi, subi, andi, ori, nori
- הוראות הסתעפות מותנית: beq, bne, blt, bgt
- הוראות טעינה ושמירה בזיכרון: lb, sb, lw, sw, lh, sh

הוראות אריתמטיות ולוגיות מסווג I:

הוראות addi, subi, andi, ori, nori מבצעות פעולה אריתמטית או לוגית בין אוגר rs לבין הערך המידי immed והותקאה מאוחסנת באוגר rt. בעט ביצוע הוראה, המעבד מרחיב את הערך ל-32 סיביות, כך שהפעולה מתבצעת על אופרנדים ברוחב 32 סיביות.

- הוראה addi מבצעת פעולה חיבור, ככלומר $rt=rs+immed$
- הוראה subi מבצעת פעולה חיסור, ככלומר $rt=rs-immed$
- הוראה andi מבצעת פעולה and בין הסיביות של האופרנדים, ככלומר $rt=rs \& immed$
- הוראה ori מבצעת פעולה or בין הסיביות של האופרנדים, ככלומר $rt=rs | immed$

- ההוראה `nori` מבצעת פעולה `nor` בין הסיביות של האופרנדים, כלומר (`rt = ~ (rs | immed)`)

נדגים את השימוש בהוראות, ואת אופן קידודן :

`addi $9, -45, $8`

הוראה זו מבצעת חיבור של תוכן האוגר `$9` עם הערך המידי `-45` ושומרת את התוצאה באוגר `$8`.
הערה : נשים לב שיש הבדל בסדר האופרנדים בין ההוראה לשפת אסמבלי לבין הקידוד הבינארי.

בקידוד הוראה זו, שדה `opcode` יכול את קוד הפעולה של `addi`, השדה `rs` יכול את מספרו של האוגר שהוא המוחבר הראשון (בדוגמה זו `9`), השדה `rt` יכול את מספרו של אוגר התוצאה (בדוגמה זו `8`), והשדה `immed` יכול את הקבוע שהוא המוחבר השני (בדוגמה זו המספר `-45`).

באופן דומה ל- `addi`, משתמשים גם בהוראות : `subi`, `andi`, `ori`, `nori`.
נדיש שובה בהוראה `subi`, המוחסר הוא האוגר `rs`, והמחסר הוא הערך המידי.
אין מניעה שאוthon אוגר ישמש בשני האופרנדים `rs`, `rt`.

הוראות הסתעפות מותניות:

ההוראות `beq`, `bne`, `blt`, `bgt` משווות בין תכני האוגרים `rs` ו-`rt`, ובודקות האם תוצאה ההשוואה מתאימה לתנאי הנתון (לפי קוד הפעולה) , ואם כן, מתבצעת קפיצה להוראה בכתובות עד שמצוינה ע"י התווית `label`.

עד הקפיצה מקודד בשדה `immed`, באמצעות המרחק אל הכתובת `label` מהכתובת הנוכחית (כתובות הוראות הסתעפות). המרחק יכול להיות חיובי או שלילי, אך אינו יכול לחרוג מגבולות של מספר ברוחב 16 סיביות בשיטות המשלימים `L-2`.

התווית שמצוינה את עד הקפיצה חייבת להיות מוגדרת בקובץ המקור הנוכחי (כלומר זו אינה תווית חיונית).

- ההוראה `beq` בודקת האם תוכן האוגר `rs` שווה לתוכן האוגר `rt`, ואם כן אז מתבצעת הקפיצה
- ההוראה `une` בודקת האם תוכן האוגר `rs` שונה מתוכן האוגר `rt`, ואם כן אז מתבצעת הקפיצה
- ההוראה `blt` בודקת האם תוכן האוגר `rs` קטן מtocן האוגר `rt`, ואם כן אז מתבצעת הקפיצה
- ההוראה `bgt` בודקת האם תוכן האוגר `rs` גדול מtocן האוגר `rt`, וכן כן אז מתבצעת הקפיצה

נדגים את השימוש בהוראות, ואת אופן קידודן :

`blt $5, $24, loop`

הוראה זו בודקת האם תוכנו של אוגר `5` קטן מtocנו של אוגר `24`, ואם כן, ההוראה הבאה שתתבצע תהיה בכתובות `loop`, ואחרת ההוראה הבאה תמשיך להיות ההוראה העוקבת.

בקידוד הוראה זו, שדה `opcode` יכול את קוד הפעולה של `blt`, השדה `rs` יכול את מספרו של האוגר המשמש כאופרנד השמאלי של פעולה ההשוואה (בדוגמה זו `5`), והשדה `rt` יכול את מספרו של האוגר המשמש כאופרנד הימני (בדוגמה זו `24`).

השדה `immed` יכול את המרחק מכתובת ההוראה הנוכחית (`blt`) אל ההוראה בכתובות `loop`. את המרחק הזה על האסמבילר לחשב בעצמו. למשל, נניח שההוראה `blt` הנוכחית נמצאת בכתובות `300`, והכתובות `loop` היא `200`. השדה `immed` יקודד לערך `-100`.

באופן דומה ל- `blt`, משתמשים גם בהוראות `bne`, `beq`, `bgt`

ההוראות טעינה ושמירה בזיכרון:

ההוראות sh, sw, lw, lh, sb, word טוונות לאוגר ערך (שלם) שנמצא בזיכרון, או שומרות בזיכרון ערך שנמצא באוגר (תליי בהוראה). גודלו של הערך שייטען/ישמר יכול להיות: byte ייחד, או word (מילה = 4 bytes), או half-word (חצי מילה = 2 bytes).

כל ההוראות האלה ניגשות כתובות בזיכרון שהיא הסכום של תוכן האוגר *rs* והערך *immed*. הקבוע *immed* משמש כאן בתפקיד של offset (היחסט) מכתובת בסיס שהיא האוגר *rs*. היחסט רשאי בתכנית בשפת אסמבלי קבוע, יוכל להיות חיובי או שלילי (במגבילות תחום הערכים ברוחב 16 סיביות בשיטת המשלים -2).

- ההוראה *lb* טוונת מהזיכרון ערך בגודל byte, אל 8 הסיביות הנמוכות (סיביות 7-0) של אוגר *rt*.
rt=Mem[rs+immed]
- ההוראה *lw* טוונת מהזיכרון ערך בגודל word, אל אוגר *rs*.
rt=Mem[rs+immed]
- ההוראה *lh* טוונת מהזיכרון בגודל half-word, אל 16 הסיביות הנמוכות (סיביות 15-0) של אוגר *rt*.
rt=Mem[rs+immed]
- ההוראה *sb* שומרת בזיכרון את 8 הסיביות הנמוכות של האוגר *rs*.
Mem[rs+immed]=rt
- ההוראה *sw* שומרת בזיכרון את תוכן האוגר *rs*.
Mem[rs+immed]=rt
- ההוראה *sh* שומרת בזיכרון את 16 הסיביות הנמוכות של האוגר *rs*.
Mem[rs+immed]=rt

הזיכרון במחשב הדמיוני שלנו מארגן בשיטת little-endian. ערך (שלם) שוגדל יותר מבית (תא זיכרון) אחד, מאוחסן בזיכרון כך שהבイト הפחות משמעותי של הערך מאוחסן בכתובת הנמוכה ביותר ברצף הבטים שתופס הערך, ואחריו שאר הבטים בסדר עולה של משמעות. כמשמעותיים לכתובת של ערך בזיכרון, זו תמיד כתובת הבית הນוק ביותר.

נדגים את השימוש בההוראות טעינה ושמירה, ואת אופן קידודן:

lh \$9, 34, \$2

הוראה זו טוונת אל האוגר \$2 שני בתים (חצי-מילה) מהזיכרון, החל מכתובת שמתකבלת מחיבור האוגר \$9 והקבוע \$34.

בקידוד הוראה זו, שדה opcode יכול את קוד הפעולה של *lh*, השדה *rs* יכול את מספר האוגר המשמש לחישוב הכתובת (בדוגמה זו \$9, האוגר *rs* יכול את מספר האוגר אליו ייטען הערך (בדוגמה זו \$2), והשדה *immed* יכול את היחסט (בדוגמה זו 34).
הערה: נשים לב שיש הבדל בסדר האופrndים, בין תחביר ההוראה בשפת אסמבלי לבין הקידוד הבינארי.

באופן דומה ל-*lh*, משתמשים גם בההוראות *lw*, *lb*,

דוגמיה נוספת:

sw \$7, -28, \$18

הוראה זו שומרת את תוכן האוגר \$18 בזיכרון, החל בכתובת שמתකבלת מחיבור של האוגר \$18 והקבוע -28.

בקידוד הוראה זו, שדה opcode יכול את קוד ההוראה של *sw*, השדה *rs* יכול את מספר האוגר המשמש לחישוב הכתובת (בדוגמה זו \$7), השדה *rs* יכול את מספר האוגר שיישמר בזיכרון (בדוגמה זו \$18), והשדה *immed* יכול את היחסט (בדוגמה זו -28).
הערה: נשים לב שיש הבדל בסדר האופrndים, בין תחביר ההוראה בשפת אסמבלי לבין הקידוד הבינארי.

באופן דומה ל-*sw*, משתמשים גם בההוראות *sh*, *sb*.

הוראות מסוג J:

הוראה מסוג J מוקודדת באופן הבא :

opcode	reg	address
31	26	25 24 0

לכל אחת מההוראות מסוג J יש opcode ייחודי (ראו טבלת ההוראות לעיל).

קיימות ארבע הוראות מסוג זה, והן : jmp, la, call, stop

הוראת jmp:

הוראה שגורמת לפיצעה למועד אחר בתוכנית לשם המשך הריצה. תחביר ההוראה jmp בשפת האסמלבי יכול להיכתב באחת משתי הצורות הבאות :

```
jmp label  
jmp $register
```

בצורה הראשונה, מתבצעת לפיצעה להוראה הנמצאת בכתובת שמצוינה ע"י התווית .label. נציג שהתווית יכולה להיות מוגדרת בקובץ המקור הנוכחי, או בקובץ מקור אחר (תוויות חיצונית).

נדגים את השימוש בצורת תחביר זו. נניח שקיים בתוכנית הוראה עם תווית main וברצוננו לקפוץ אליה, אז ניתן :

```
jmp main
```

בקידוד הוראה זו לשפת מכונה, נציב בשדה ה- opcode את קוד הפעולה של jmp (ראו טבלת הפעולות לעיל). בשדה poz נציג 0, כדי לציין שיעד הקפיצה נתון באמצעות תוויות. אם התווית main מוגדרת בקובץ המקור הנוכחי (איינה חיצונית) אז נציב בשדה ה- address את כתובתה (סיבית 0 בקידוד היא הסיבית הפחות משמעותית של הכתובת). אם התווית היא חיצונית, אז נציב בשדה ה- address אפסים, כי הכתובת האמיתית לא ידועה לאסמלבלר (קידוד השדה יישלם בשלב הקישור של תהליך בנין התוכנית, שאותו איןכים נדרש ממש).

בצורת התחביר השנייה של הוראת jmp מתבצעת לפיצעה לכתובת הנמצאת באוגר, שמספרו מוקוד בשדה address (כלומר address יכול מספר בין 0-31 ברוחב 5 סיביות). הערת : מרחיב הכתובות במוחשב הדמיוני הוא התהום [1..2²⁵-1], והאוגר אמרור להכיל ערך שאינו חורג מתחום זה.

נדגים את השימוש בצורת תחביר זו. נניח שאוגר 7 מכיל כתובת שאליה נרצה לקפוץ, אז ניתן :

```
jmp $7
```

בקידוד הוראה זו לשפת מכונה, נציב בשדה ה- opcode את קוד הפעולה של jmp. בשדה reg נציג 1, כדי לציין שיעד הקפיצה נתון באמצעות אוגר. בשדה address נציב את מספר האוגר (בדוגמה זו 7). למעשה, סיביות 4-0 יכilio את מספר האוגר, וסיביות 5-24 יכilio אפסים.

הוראת la:

הוראת la (load address) טעונה לאוגר 0 כתובת של תווית מסוימת. נציג שהתווית יכולה להיות מוגדרת בקובץ המקור הנוכחי, או בקובץ מקור אחר (תוויות חיצונית).

תחביר ההוראה la בשפת האסמלבי הוא :

```
la label
```

label מצין את התווית שאת כתובתה מעוניינים לטעון לאוגר 0\$.
הערה : לא ניתן בהוראה `la` להשתמש באוגר אחר מאשר 0\$.

נדגים את השימוש בהוראה זו. נניח שבזיכרנו מוגדר משתנה בשם `num1` שמשתנה שנמצא בכתובת `num1`, ואנו מעוניינים לטעון את כתובת המשתנה לאוגר 0\$, אזי נכתב：
`la num1`

בקידוד הוראה זו לשפט מכונה, נציב בשדה ה- `opcode` את קוד הפעולה של `la`. השדה `reg` אינו בשימוש בהוראה זו, ולכן יוצב בו תמיד 0. אם התווית `num1` מוגדרת בקבץ המקור (אינה חיונית) אזי נציב בשדה `address` את כתובתה. אם התווית היא חיונית, אזי נציב בשדה `address` אפסים (עם אותה משמעות כמו בהוראת `jmp`).

הוראת call:

הוראה שגורמת לפיצח למקום אחר בתכנית לשם המשך הריצה, ושמורת את כתובת ההוראה הבאה (ההוראה העוקבת אחרי `call`). ההוראה משמשת בעיקר לקרוא לשגרה. תחביר הוראה זו בשפת האסמבלי הוא :

`call label`

מתבצעת פיצח להוראה הנמצאת בכתובת שמצוינת ע"י התווית `label`, ואילו בכתובת ההוראה העוקבת ל- `call` נשמרת באוגר 0\$. נDIGISH שהתוויות יכולה להיות מוגדרת בקובץ המקור הנוכחי, או בקובץ מקור אחר (תוית חיונית).

הערה : לא ניתן בהוראה `call` להשתמש באוגר אחר מאשר 0\$.

נדגים את השימוש בהוראה זו. נניח שקיים בתכנית הוראה עם תוית `myfunc`, שהיא ההוראה הראשונה של שירה, וברצוננו לקרוא לשגרה. אזי נכתב：
`call myfunc`

בקידוד הוראה זו לשפט מכונה, נציב בשדה ה- `opcode` את קוד הפעולה של `call`. השדה `reg` אינו בשימוש בהוראה זו, ולכן יוצב בו תמיד 0. אם התווית `myfunc` מוגדרת בקבץ המקור (אינה חיונית) אזי נציב בשדה `address` את כתובתה. אם התווית היא חיונית, אזי נציב בשדה `address` אפסים (עם אותה משמעות כמו בהוראת `jmp`).

כאמור, כתובת ההוראה שאחרי `call` (כתובת החזרה מהשגרה) נשמרת באוגר 0\$. כדי לחזור מהשגרה, יש להשתמש בהוראה `jmp` בצורתה השנייה (ראה לעיל)

הוראת stop:

להוראה זו אין אופרנדים והוא נועדה לעצור את התוכנית. בקידוד הוראה זו לשפט מכונה, נציב בשדה ה- `opcode` את קוד הפעולה של `stop`, ובכל יתר השדות יוצבו אפסים.

מבנה תכנית בשפת אסמבלי :

תכנית בשפת אסמבלי בנויה ממשפטים (statements). קובץ מקור בשפת אסמבלי מורכב משורות המכילות משפטי של השפה, כאשר כלמשפט מופיע בשורה נפרדת. כלומר, ההפרדה בין משפט למשפט בקובץ המקור הינה באמצעות התו 'ט' (שורה חדשה).

אורכה של שורה בקובץ המקור הוא 80 תוים לכל היתר (לא כולל התו טו).

יש ארבעה סוגי משפטיים (שורות בקובץ המקור) בשפת אסמבלי, והם :

סוג המשפט	הסביר כללי
משפט ריק	זהוי שורה המכילה אך ורק תווים לבנים (whitespace), ככלומר רק רווחים וטאים. ייתכן ובשורה אין אףתו (למעט התו 'ו'), ככלומר השורה ריקה.
משפט הערתה	זהוי שורה בה התו הראשון שאינו תוו 'ו' (נקודה-פסיק). על האסמלר להתעלם לוחלטן משורה זו.
משפט הנטהיה	זהו משפט המנחה את האסמלר מה עליו לעשות כשהוא פועל על תוכניתה המקור. יש מספר סוגים של משפטי הנטהיה. משפט הנטהיה עשוי לגרום להקצת זיכרונו ואתחול משתנים של התוכנית, אך הוא אינו מייצר קידוד של הוראות מכונה המיועדות לביצוע בעת ריצת התוכנית.
משפט הוראה	זהו משפט המייצר קידוד של הוראות מכונה לביצוע בעת ריצת התוכנית. המשפט מורכב ממשם ההוראה (פעולה) שעל המעבד לבצע, והאפרנדים של ההוראה.

cut נפרט יותר לגבי סוגי המשפטים השונים.

משפט הנטהיה:

משפט הנטהיה הוא בעל המבנה הבא :

בתחילת המשפט יכול להופיע הגדרה של תווית (label). לתווית יש תחביר חוקי שיתוואר בהמשך. התווית היא אופציונאלית.

לאחר מכן מופיע שם הנטהיה. לאחר שם הנטהיה יופיע פרמטרים (מספר הפרמטרים בהתאם להנטהיה).

שם של הנטהיה מתחילה בטעו ' ' (נקודה) ולאחריו תווים באותיות קטנות (lower case) בלבד.

יש כמה סוגים (שמות) של משפטי הנטהיה, והם :

1. משפטי הנטהיה 'db' , 'dw' , 'dd' , ' .dw' , ' .db' , ' .dd' .

הפרמטרים של משפטי הנטהיה 'db' , 'dw' , 'dd' , ' .dw' , ' .db' , ' .dd' הם מספרים שלמים חוקיים (אחד או יותר) המופרדים על ידי הטו ' ' (פסיק). לדוגמה :

.db 7, -57, 17, +9

.dw 120056

.dh 0, -60431, 1700, 3, -1

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

משפט הנטהיה אלה מנחים את האסמלר להקצות מקום בתמונה הנתונים (data image), שבו יאוחסנו הערכים של הפרמטרים, ולאחר מכן המנצח את מונה הנתונים, בהתאם למספר הערכים ובהתאם לגודלם. כל נתון המוגדר ע"ימשפט הנטהיה 'db' , תופס בית (byte) בודד. נתון המוגדר ע"י משפט הנטהיה 'dw' , תופס ארבעה בתים (מילה). נתון המוגדר ע"י 'dh' , תופס שני בתים (חצי-מילה).

הערה : נשים לב שככל מספר חייב להיות בגודל מתאים, שלא יתרוג מגבלות הייצוג של סוג הנתון המוגדר.

אם בהנחיה מוגדרת תווית, אז תווית זו מקבלת את ערך מונה הנתונים (לפni הקידום), ומוכנסת אל טבלת הסמלים. דבר זה מאפשר להתייחס אל מקום מסוים בתמונה הנתונים דרך שם התווית (למעשה, זהה דרך להגדיר שם של משתנה).

לדוגמה, אם נכתב :

XYZ: .dh 0, -60431, 1700, 3, -1

אז יוקצו בתמונה הנתונים 10 בתים (bytes) רצופים, וכל שני בתים (חצי-מילה) מאותחלים עם אחד המספרים שופיעו בhhnnhh (לפי הסדר). התווית XYZ מזוהה עם כתובות הבית הראשון. אפשר להתייחס ל-XYZ כמשנה שהוא מערך של 5 איברים בגודל חצי-מילה.

2. ההנחיה 'asciz'

להנחיה 'asciz', פרמטר אחד, שהוא מחרוזת חוקית. תוווי המחרוזת מקודדים לפי ערכי ה-ascii המתאים, ומוכנסים אל תמונה הנתונים לפי סדרם, כל TWO bytes. בסוף המחרוזת יתווסף התו '0' (הערך המספרי 0), המסמן את סוף המחרוזת. מונה הנתונים של האסמביל יקודם בהתאם לאורך המחרוזת (ב途וספת מקום אחד עבור התו המסיים). אם בשורת ההנחיה מוגדרת תווית, אז תווית זו מקבלת את ערך מונה הנתונים (לפni הקידום) ומוכנסת אל טבלת הסמלים.

לדוגמה, אם נכתב :

STR: .asciz "hello world"

אז יוקצו בתמונה הנתונים 12 בתים רצופים, שיואתחלו לתוווי המחרוזת לעיל (כולל האפס המסיים). התווית STR מזוהה עם כתובות התו הראשון. אפשר להתייחס ל-STR כמשנה שהוא מערך (מקסימלי) 11.

3. ההנחיה 'entry'

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

לדוגמה, השורות :

entry HELLO
HELLO: add \$1,\$2,\$5

מודיעות לאסמביל שייתכן ותהייה התייחסות בקובץ מקור אחר לתווית HELLO המוגדרת בקובץ הנוכחי.

ltshomat lib: תווית המוגדרת בתחילת שורת entry. הינה חסרת משמעות והאסמביל מועלם מתווית זו (אפשר שהאסמביל יוציא הודעה נוספת אזהרה).

4. ההנחיה 'extern'

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

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

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

.extern HELLO

لتשומת לב: לא ניתן להגיד באותו הקובץ את אותה התווית גם כ-entry וגם כ-extern (בדוגמאות לעיל, התווית HELLO). כל הרגע הוא לאפשר להוראה בקובץ אחד להשתמש בתנונים מקובץ אחר, או לפחות להוראה בקובץ אחר.

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

משפט הוראה:

משפט הוראה מורכב מחלקים הבאים:

1. תווית אופציונלית.
2. שם הפעולה.
3. אופרנדים, בהתאם לסוג הפעולה (כפי שהסביר לכל פעולה לעיל)

אם מוגדרת תווית בשורת ההוראה, אז היא תוכנס אל טבלת הסמלים. ערך התווית יהיה מען ה-byte הראשון של ההוראה בתוך תמונה הקוד שבונה האסמבלר.

שם הפעולה ייכתב תמיד באותיות קטנות (lower case). לאחר שם הפעולה יופיעו האופרנדים, בהתאם לסוג הפעולה. יש להפריד בין שם-הפעולה לבין האופרד הראשון באמצעות רווחים ו/או טאים (אחד או יותר).

כאשר יש מספר אופרנדים, האופרנדים מופרדים זה מזה בטו' (פסיק). **לא חייבת להיות הצמדה של האופרנדים לפסיק.** כל כמהות של רווחים ו/או טאים משנה צידי הפסיק היא חוקית.

ಅಫ್ಯಾನ ಶಿದ್ಧತೆ ಬಹಿರಳಿಗೆ ಶಂಕಾ ಅಸಮಿಲಿ

תווית:

תווית היא סמל שמודדר בתחילת משפט הוראה או בתחילת המגדיר משתנים. תווית חוקית מתחילה באות אלפביתית (גדולה או קטנה), ולאחריה סדרה כלשהי של אותיות אלפביתיות (גדולות או קטנות) ו/או ספרות. האורך המקסימלי של תווית הוא 31 תווים.

הגדרה של תווית מסוימת בטו': (נקודותים).תו זה אינה מהויה חלק מהתווית, אלא רק סימן המציין את סוף ההגדרה. התו ':' חייב להיות צמוד לתווית (ללא רווחים).

אסור שאותה תווית תוגדר יותר מפעם אחת (כموון בשורות שונות). אותיות קטנות וגדלות ㄣಚಿಬ್ಬತೆ ಶಿವಾನ್ಯಾತ್ಮಾ ಮಾನ್ಯ.

לדוגמה, התוויות המוגדרות להלן הן תוויות חוקיות.

hEllo:

x:

He78902:

لتשומת לב: מילים שמורות של שפת האסמבלי (כלומר שם של פעולה או הינה) אינן יכולות לשמש גם כשם של תווית. לדוגמה: ה слова `add`, `asciz` לא יכולים לשמש כתוויות, אבל ה слова `ASCiz`, `Add` הן תוויות חוקיות.

התווית מקבלת את ערכה בהתאם להקשר בו היא מוגדרת. תווית המוגדרת בשורת הינה מקבלת את ערך מונה הנתונים (data counter) הנוichi, בעוד שתווית המוגדרת בשורת הוראה מקבלת את ערך מונה ההוראות (instruction counter) הנוichi.

لتשומת לב: מותר במשפט הוראה להשתמש באופרנד שהוא תווית המוגדרת בשורה יותר מאוחרת בקובץ המקור. כמו כן, מותר במשפט הוראה להשתמש באופרנד שהוא סמל שאינו מוגדר כתווית בקובץ הנוichi, כל עוד הסמל מופיע בchiezoni (באמצעות הניתнт exern). בקובץ הנוichi. נושא זה יובחר בהמשך, בדיעון על מימוש האסמבלי.

מספר:

מספר חוקי מתחילה בסימן אופציוני: ‘–’ או ‘+’, ולאחריו סדרה של ספרות בסיס עשרוני. לדוגמה: `+123`, `-576`, הם מספרים חוקיים. אין תמיכה בשפת האסמבלי שלנו בייצוג בסיס אחר מאשר עשרוני, ואין תמיכה במספרים שאינם שלמים.

מחרוזות:

מחרוזת חוקית היא סדרת תווים `ascii` נראה (שניטנים להדפסה), המוקפים במרכאות כפולות (המרכאות אינן נחשבות חלק מהמחרוזות). דוגמה למחרוזת חוקית: “hello world”.

אסמבלי עם שני מעבריים

האסמבלי מקבל כקלט קובץ המכיל תוכנית בשפת אסמבלי, ותפקידו להמיר את קוד המקור לקוד מכונה, ובנותו קובץ פלט המכיל את קוד המכונה.

הערה: האסמבלי לא מבצע את התוכנית, אלא רק מתרגם אותה לבינארי. לפיכך, על קוד המקור להיות נכון מבחינה תחבירית, אבל זה לא משנה לאסמבלי (ולפרויקט שלנו) מה התוכנית אמרה לעשות והאם היא בכלל תעבוד נכון.

דוגמה: האסמבלי מקבל את התוכנית הבאה בשפת אסמבלי:

```
MAIN: add    $3,$5,$9
      ori    $9,-5,$2
            la     val1
            jmp   Next
      Next: move   $20,$4
            bgt   $4,$2,END
            la     K
            sw    $0,4,$10
            bne   $31,$9,LOOP
            call   val1
            jmp   $4
      END: stop
      STR: .asciz "aBcd"
      LIST: .db    6,-9
             .dh    27056
             .entry K
             K:    .dw    31,-12
             .extern val1
```

התרגומים של תוכנית המקור שבדוגמה לקוד מכונה מוצג להלן. לכל שורה בקוד המקורי, מוצג הקידוד הבינארי של הוראה או של הנתונים, בצירוף הכתובת בזיכרון לשם ייטען הקוד לצורך הרצה.

האסמבילר בונה את קוד המcona של התוכנית כך שיתאים לטעינה בזיכרון **חט' מכתובת 100** (עשרהוני).

שיםו לב לKENCODES ב-4 בעמודת הכתובת בכל פעם שמקודדים הוראה, לאחר והקידוד מרכיב מס' 32 סיביות, שתופסים 4 תא זיכרון רצופים.

Address (decimal)	Source Code	Machine Code (binary)
0100	MAIN: add \$3,\$5,\$9	000000 00011 00101 01001 00001 000000
0104	LOOP: ori \$9,-5,\$2	001101 01001 00010 11111111111111011
0108	la val1	011111 0 00000000000000000000000000000000
0112	jmp Next	011110 0 000000000000000000000000000000001110100
0116	Next: move \$20,\$4	000001 10100 00000 00100 00001 000000
0120	bgt \$4,\$2,END	010010 00100 00010 00000000000011000
0124	la K	011111 0 00000000000000000000000010011101
0128	sw \$0,4,\$10	010110 00000 01010 0000000000000000100
0132	bne \$31,\$9,LOOP	001111 11111 01001 111111111100100
0136	call val1	100000 0 00000000000000000000000000000000
0140	jmp \$4	011110 1 00000000000000000000000000000000100
0144	END: stop	111111 0 00
0148	STR: .asciz "aBcd"	01100001
0149		01000010
0150		01100011
0151		01100100
0152		00000000
0153	LIST: .db 6,-9	00000110
0154		11110111
0155	.dh 27056	0110100110110000
0157	K: .dw 31,-12	0000000000000000000000000000000011111
0161		111111111111111111111111111111110100

נთאר עתה כיצד האסמבילר יבצע את התרגומים לקוד מכונה באופן שיטתי.

האסמבילר מחזיק טבלה שבה רשומים כל שמות הפעולה של ההוראות והקודים הבינאריים (opcode, funct) המתאימים להם, בדומה לטבלה שהציגנו קודם. לכן קל לתרגם לבינארי את שם הפעולה : כשקראת שורת הוראה מסוימת המקורי, מחפשים בטבלה את הקוד הבינארי המתאים.

גם אופרנדים שהם אוגרים או קבועים מיידיים, אפשר לקודד ישירות לבינארי. כך גם לגבי נתונים שמוגדרים ומאותחלים באמצעות שורות הנקה.

הcosaוי העיקרי הוא לקודד לבינארי אופרנדים המשתמשים בסמלים (תוויות). על האסמבילר לדעת מה ערך הnumerical (הכתובת) שמייצג כל סמל. אולם בהבדל מהקודים של שמות הפעולה, שהם ערכיהם קבועים וידועים, הרי הכתובת בזיכרון של הסמלים שבסימוש התוכנית לא ידועות, עד אשר תוכנית המקור נסרקה כולה ונתגלו כל הגדרות הסמלים.

למשל, בקוד המקורי לעיל, האסמבילר אינו יכול לדעת שהסמל END-Amor להיות משוויך לכתובת 144 (עשרהוני), ושהסמל K-Amor להיות משוויך לכתובת 157, אלא רק לאחר שנקראו כל שורות התוכנית.

לכן מפרידים את הטיפול של האסטובלר לשני שלבים. בשלב הראשון מזוהים את כל הסמלים שבתכנית המקור וקובעים מהם הערכיים המספריים המשווים להם. בשלב השני מחליפים את הסמלים שבאופרנדים של הוראות התוכנית בערכיהם המספריים, וכך ניתן לקודד את האופרנדים לבניאר.

מעבר ראשון, האסמבלר בונה טבלת סמלים (symbol table), ומכניס לתוכה כל סמל שМОדר בתוכנית המקור, יחד עם הערך המספרי (המשמעותי) המשוויק לו.

מעבר שני, האסמבller משתמש בטבלת הסמלים כדי להמיר את השורות בקוד המקור לקוד מכונת. נשים לב שבתחילת המעבר השני צרכיים העריכים של כל הסמלים להיות כבר ידועים. אם יש סמל שנעשה בו שימוש באופרנד, אבל הסמל לא נמצא בטבלת הסמלים, לא ניתן להשלים את הקידוד.

עבור הדוגמה שהציגו, טבלה הסמלים נתונה להלן. לכל סמל יש בטבלה גם מאפיינים (attributes) שיווסבו בהמשך. אין חשיבות לסדר השורות בטבלה (כאן הטבלה לפי הסדר בו הוגדרו הסמלים בתכנית המקור).

Symbol	Value (decimal)	Attributes
MAIN	100	code
LOOP	104	code
Next	116	code
END	144	code
STR	148	data
LIST	153	data
K	157	data, entry
val1	0	external

לתשומת לב: כאמור, תפקיד האסטובלר, על שני המUberים שלו, לתרגם קוד מקור לקוד בשפת מוכונה. בגמר פועלות האסטובלר, התוכנית טרם מוכנה לטעינה ל זיכרון לצורך ביצוע. קוד המוכונה חייב לעبور לשלב הקיורו/טעינה, ו록 לאחר מכן לשלב הביצוע (שלבים אלה אינם חלק מהממען').

המעבר הראשון

במעבר הראשון נדרשים כללים כדי לקבוע איזה מען ישוק לכל סמל. העיקרון הבסיסי הוא לספר את המקומות בזיכרון, אותן תופסות ההוראות. אם כל הוראה תיתען בזיכרון למקום העוקב להוראה הקודמת, תציין ספריה זאת את מען ההוראה הבא. הספריה נעשית על ידי האסטבלר ומוחזקת במונח ההוראות (IC). ערכו ההתחלתי של IC הוא 100 (עשרה), ולכן קוד המכונה של ההוראה הראשונה נבנה כך שייתען לזכור החל ממען 100. ה-IC מוגדל ב-4 אחרי כל שורת הוראה (כמספר התאים שתופס קידוזה ההוראה). כך ה-IC מבצע תמיד על המיקום הפנוי הבא בזיכרון.

כמו כן, באופן דומה, האסembler סופר את המיקומות בזיכרון שתופסים הנתונים (המודדרים באמצעות הנקודות). הספירה מוחזקת במונה הנתונים (DC). (ראו פרטים נוספים בהמשך, בדיעו על תהליך העבודה של האסmbler).

כאמור, כדי לקודד את החראות בשפת מכונה, מחזיק האסמבילר טבלה, שיש בה לכל פעללה קידוד מותאים (opcode), ובחלק מהפעולות גם (funct). בזמן התרגום לקוד מכונה, מחליף האסמבילר כל שם פעללה בקידוד שלה.

כasher natakl haasembler battoiot hampiyyah batchilit hashorah, hoa yod'uh shlefpiyoh hagdara shel tuoiyut, ve'oz hoa meshikh la muu - urevo haonchi shel h-IC avo h-DC. CK mukbelot cel tuoiyut at meuna boshorah ba hia mogderet. tuoiyut mongesht l'sebelat h'smelim. hamkilla bnosoy le'sham tuoiyut gom at hamzon

ומאפיינים נוספים. כאשר יש התייחסות לתוויות באופrnd של הוראה כלשהי, יוכל האסמבולר לשולף את המען המתאים מטבלת הסמלים.

הוראה יכולה להתיחס גם לסדר שטרם הוגדר עד כה בתוכנית, אלא יוגדר רק בהמשך התוכנית.
לדוגמה, להלן הוראת קפיצה למען שימושי לתוית A, אך התוית מוגדרת רק בהמשך הקוד :

```
jmp A  
.  
.  
.  
A: .....
```

כאשר מגיע האסמבולר להוראת הקפיצה (jmp), הוא טרם נתקל בהגדרת התוית A וכמובן לא ידוע את המען המשויך לתוית. אך האסמבולר לא יכול לבנות את הקידוד הבינארי של האופrnd A. ראו בהמשך כיצד נפתרה בעיה זו.

בכל מקרה, תמיד אפשר כבר במעבר הראשון לבנות באופן חלקית את קוד המכונה של ההוראה. למשל, אפשר לקודד את שדה ה-opcode, את שדה ה-funct (בהוראה בפורמט R), וכן את כל השדות המכילים מספרי אוגרים. בהוראות אריתמטיות/לוגיות בפורמט I אפשר לקודד במעבר ראשון גם את השדה .immed.

המעבר השני

ראינו שבמעבר הראשון, האסמבולר אינו יכול לבנות את קוד המכונה של אופrndים המשתמשים בסמלים שעדיין לא הוגדרו. רק לאחר שהאסמבולר עבר על כל התוכנית, כך שכל הסמלים נקבעו כבר לטבלת הסמלים, יוכל האסמבולר להשלים את קוד המכונה.

לשם כך מבצע האסמבולר מעבר נוסף (מעבר שני) על כל קובץ המקור, ובונה את קוד המכונה של כל אופrnd שהוא ערך של סמל, באמצעות ערכי הסמלים בטבלת הסמלים.

ספקטיבית, יש לעדכן את שדה ה-.immed בהוראות הסטעפות מותנית מסוג I, ואת שדה הכתובת בהוראה מסוג J.

בסוף המעבר השני, תהיה התוכנית מתרגמת בשלמותה לקוד מכונה.

הפרדת הוראות ונתונים

בתוכנית מבחןים בשני סוגים של תוכן : הוראות ונתונים. יש לאorgan את קוד המכונה כך שתהייה הפרדה בין הנתונים וההוראות. הפרדת ההוראות והנתונים לקטעים שונים בזיכרון היא שיטה עדיפה על פני הצמדה של הגדרות הנתונים להוראות המשמשות בהן.

אחת הסכנות הטموנותabei הפרדת ההוראות מהנתונים היא, שלפעמים עלול המעבד, בעקבות שגיאה לוגית בתוכנית, לנסות "לבצע" את הנתונים כailo הוי הוראות חוקיות. למשל, שגיאה שיכולה לגרום תופעה כזו היא הסטעפות לא נכונה. התוכנית כMOVן לא תעבד נכון, אך לרוב הנזק הוא יותר חמור, כי נוצרת חריגת חומרה ברגע שהמעבד מבצע פעולה שאינה חוקית.

האסמבולר שלנו חייב להפריד, בקוד המכונה שהוא מיצר, בין קטע הנתונים לקטע ההוראות. כמובן **בקובץ הפלט (בקוד המכונה) תהיה הפרדה של הוראות ונתונים לשני קטיעים נפרדים**, אם כי **בקובץ הקלט אין חובה שתהייה הפרדה זו**. בהמשך מתואר אלגוריתם של האסמבולר, ובו פרטיים כיצד לבצע את הפרדה.

גילוי שגיאות בתוכנית המקור

על האסמלר לבצע ניתוח תחבירי (parsing) מדויק של תכנית המקור בשפת אסמלרי, ולהיות מסוגל לגלוות ולדוח על מגוון שגיאות, כגון: שם-פעולה שאינו מוגדר, מספר אופרנדים שגוי, סוג אופרנד שאינו מתאים לפעולה, מספר אוגר שלא קיים, שימוש בסמל שאינו מוגדר, סמל שמודר יותר מפעם אחת, ערך מספרי בגודל חריג, ועוד שגיאות אחרות.

האסמלר ידוח על השגיאות באמצעות הודעות אל הפלט הסטנדרטי `stdout`. כל שגיאה נגרמת (בדרך כלל) על ידי שורה מסוימת בקוד המקור. בכל הودעת שגיאה יש לפחות גם את מספר השורה בה זוהתה השגיאה (מנין השורות בקובץ מתחילה -1).

דוגמה: אם בשורה מס' x בקובץ הקלט מופיעים שני אופרנדים בהוראה שאמור להיות בה רק אופרנד אחד, האסמלר ייתן הודעת שגיאה בנוסח "Line x: extraneous operand".

דוגמה: אם בשורה מס' y בקובץ הקלט מופיע שם לא מוכר של הנחיה, האסמלר ייתן הודעת שגיאה בנוסח "Line y: unrecognized directive <directive-name>".

لتשומת לב: האסמלר אינו עוצר את פעולתו אחרי שנמצאה השגיאה הראשונה, אלא ממשיך לעبور על הקלט כדי לגלוות שגיאות נוספות, ככלISION. כמובן שאין כל טעם לבנות את קבצי הפלט אם נתגלו שגיאות (ممילא אי אפשר להשלים את קוד המcona).

תהליך העבודה של האסמלר

נתאר כתוב פרוטרוט את תהליך העבודה של האסמלר. בהמשך, יוצג אלגוריתם שלו למעבר ראשון ושני.

האסמלר מתחזק שני מערכיים, שייקראו להלן תMOVה ההוראות (code) ותMOVה הנתונים (data). מערכיים אלו נותנים למשנה תMOVה של זיכרון המcona (כל איבר במערך הוא בגודל byte). במערך ההוראות בונה האסמלר את הקידוד של הוראות המcona שנקרוואו במהלך המעבר על קובץ המקור. במערך הנתונים מכניס האסמלר את קידוד הנתונים שנקרוואו מקובץ המקור (כלומר הנתונים בשורות הנחיה מסוג `.dh`, `.dw`, `.db`).

האסמלר משתמש בשני מונחים, שנקרואים IC (MOVה ההוראות - Instruction-Counter) ו- DC (MOVה הנתונים - Data-Counter). מונחים אלו מצביעים על המיקום הבא הפניו במערך ההוראות ובמערך הנתונים, בהתאם. בכל פעם כשמוחלט האסמלר לעبور על קוד מקור, המונה IC מקבל ערך התחלתי 100, והמונה DC מקבל ערך התחלתי 0. הערך ההתחלתי IC=100 נקבע כדי שקוד המcona של התוכנית יתאים לטעינה לזכרון (לצורך ריצה) החל מכתובת 100.

בנוסף, מתחזק האסמלר טבלה, אשר בה נאספות כל התווויות בהן נתקל האסמלר במהלך המעבר על קובץ המקור. לטבלה זו קוראים טבלת-הסמלים (symbol-table). לכל סמל נשמרם בטבלה שם הסמל, ערכו המספרי, ומאפיינים נוספים (אחד או יותר), כגון המיקום בתMOVה הזיכרון (data) או code, וסוג הנראות של הסמל (entry או external).

במעבר הראשון האסמלר בונה את טבלת הסמלים, ואת השלד של תMOVה הזיכרון (מערך ההוראות ומערך הנתונים לחוד).

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

1. שורה ריקה או שורת הערת:

האסטבלר מתעלם מהשורה ועובד לשורה הבאה.

2. שורת הוראה:

האסטבלר מנתח את השורה ומפענח מהי ההוראה.

אם האסטבלר מוצא בתחילת שורת ההוראה גם הגדרה של תווית, אזי התווית (הסמל) המוגדרת מוכנסת לטבלת הסמלים. ערך הסמל בטבלה הוא IC, והמאפיין הוא code

האסטבלר מכניס לתמונה הזיכרון שורה עבור ההוראה שנקרה, ובה כתובות IC, והשדות של קוד המכונה בהתאם לסוג ההוראה (J, I, R). כל שדה שניtinן לקוד כבר עתה, יקבל את הקידוד המתאים (ראו תאור המעבר הראשון לעיל).

לבסוף, האסטבלר מוסיף 4 לערכו של IC, לאחר וכל הוראה תקודד ל-4 תא זיכרון (32 סיביות).

3. שורת הנחיה:

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

I. הנחיה .dh .dw .db .

האסטבלר קורא את רשימת המספרים, מכניס כל מספר אל מערך הנתונים (בקידוד בינארי) ובכמויות בתים בהתאם לסוג משפט ההנחיה. ומקדם את מצביע הנתונים DC ב-בכמויות הבטים הכוללת שכל המספרים צרכו.

אם בשורה 'data', מוגדרת גם תווית, אזי התווית מוכנסת לטבלת הסמלים. ערך התווית הוא ערך מונה הנתונים DC שלפניהם הכנסת המספרים למערך. המאפיין של התווית הוא data.

II. הנחיה .asciz

הטיפול ב-.asciz, דומה ל-'db', אלא שקודם ה-ascii של התווים הם אלו המוכנסים אל מערך הנתונים (כל تو ב-byte נפרד). לבסוף מוכנס למערך הנתונים התו '\0' (המצין סוף מחירות). המונה DC מוקודם באורך המחרוזת + 1 (גם התו המסיים את המחרוזת יופס מקום).

הטיפול בתווית המוגדרת בהנחיה 'asciz', זהה לטיפול הנעשה בהנחיות db dw dh .

III. הנחיה .entry

זהוי הנחיה לאסטבלר לאפיין את התווית הנתונה כאופrndכ-entry בטבלת הסמלים. בעת הפיקת קבצי הפלט (ראז בהמשך), התווות תירשם בקובץ ה-entries.
לתשומת לב: זה לא נחשב כשיינא אם בקובץ המקור מופיעה יותר מהנחיה entry. אחת עם אותה תווית כאופrnd. המופעים הנוספים אינם מוסיפים דבר, אך גם אינם מפריעים.

IV. הנחיה .extern

זהוי הצהרה על סמל (תווית) המוגדר בקובץ מקור אחר, והקובץ הנוכחי עושה בו שימוש. האסטבלר מכניס את הסמל המופיע כאופrnd לטבלת הסמלים, עם הערך 0 (הערך האמייני לא ידוע, וייקבע רק בשלב הקישור), ועם המאפיין external. לא ידוע באיזה קובץ נמצא הגדרת הסמל, ואין זה רלוונטי עבור האסטבלר.

לתשומת לב: זה לא נחשב כשיינא אם בקובץ המקור מופיעה יותר מהנחיה extern. אחת עם אותה תווית כאופrnd. המופעים הנוספים אינם מוסיפים דבר, אך גם אינם מפריעים.

لتשומת לב: באופrnd של הוראה, מותר להשתמש בסמל אשר יוגדר בהמשך הקובץ (אם לאופן entry על ידי הגדרת תווית, ואם באופן עקיף על ידי הנחיתות .extern). באופrnd של הנחיתת תווית מותר להשתמש בסמל אשל יוגדר בהמשך הקובץ על ידי הגדרת תווית

בסוף המעבר הראשון, האסמבולר מעדכן בטבלת הסמלים כל סמל המופיעין כ-`data`, על ידי הוספת `(100) + IC` (ערכו של הסמל). הסיבה לכך היא שבתמונה הכלולה של קוד המכונה, תמונה הנתונים מופרדת מתמונה הוראות, וכל הנתונים נדרשים להופיע בקוד המכונה אחרי כל ההוראות. סמל מסווג `data` הוא תווית בתמונה הנתונים, והערכו מוסיף לערך הסמל (כלומר לכתובתו בזיכרון) את האורך הכללי של תמונה ההוראות, בתוספת כתובות התחלת הטעינה של הקוד, שהיא 100.

טבלת הסמלים מכילה כעת את ערכי כל הסמלים הנחוצים להשלמת תמונה הזיכרון (למעט ערכים של סמלים חיצוניים).

במעבר השני, האסמבולר משלים באמצעות טבלת הסמלים את קידוד כל ההוראות שטרם קודדו באופן מלא במעבר הראשון.

אלגוריתם שלדי של האסמבולר

לחידוד ההבנה של תהליך העבודה של האסמבולר, נציג להלן אלגוריתם שלדי למעבר הראשון ולמעבר השני.

لتשומת לב: אין חובה להשתמש דווקא באלגוריתם זה.

כאמור, אנו מחלקים את תמונה קוד המכונה לשני חלקים: תמונה ההוראות (code), ותמונה הנתונים (data). לכל חלק נתזקק מונה נפרדת: IC (מונה ההוראות) ו-DC (מונה הנתונים).

בנייה את קוד המכונה כך שייתאים לטיענה לזכרון החל מכתובת 100.

בכל מעבר מתחילה לקרוא את קוד המקור מההתחלתה.

מעבר ראשון

1. אתחל $.DC \leftarrow 0$, $IC \leftarrow 100$.
2. קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עברו ל-17.
3. אם זהה שורת הערת או שורה ריקה, חזרו ל-2.
4. האם השדה הראשון בשורה הוא תווית? אם לא, עברו ל-6.
5. הדלק דגל "יש הגדרת סמל".
6. האם זהה שורת הנחיה לאחסון נתונים, כלומר, האם הנחיתת `dh`, `dw`, `db`, `asciz`? אם לא, עברו ל-9.
7. אם יש הגדרת סמל (תווית), הכנס אותו לטבלת הסמלים עם המאפיין `data`. ערך הסמל יהיה DC . (אם הסמל אינו תווית חוקית, או שהסמל כבר נמצא בטבלה, יש להודיע על שגיאת).
8. זהה את סוג הנתונים, וקודד אותו בתמונה הזיכרון של הנתונים. רשום בתמונה הנתונים גם את הערך הנוכחי של מונה הנתונים, DC, כתובות הזיכרון של (הבית הראשון של) הנתונים שהוגדרו בשורה הנוכחית. לאחר מכן, הגדלת DC על ידי הוספת האורך הכללי של הנתונים שהוגדרו. חזרו ל-2.
9. האם זו הנחיתת `extern`. או הנחיתת `entry`? אם לא, עברו ל-12.
10. אם זהה הנחיתת `entry`. חזרו ל-2 (ההנחיתת תטופל במעבר השני).

11. אם זו החלטת *external*, הכנס את הסמל המופיע כאופרנד של ההחלטה לתוך טבלת הסמלים עם הערך 0, ועם המאפיין *external*. (אם הסמל אינו תווית חוקית, או שהסמל כבר נמצא בטבלה ללא המאפיין *external*, יש להודיע על שגיאה). חזור ל-2.
12. זוהי שורת הוראה. אם יש הגדרת סמל (תווית), הכנס אותו לטבלת הסמלים עם המאפיין *code*.
13. חפש את שם הפעולה בטבלת שמות הפעולות, ואם לא נמצא, אז הודיע על שגיאה.
14. נתח את מבנה האופרנדים של ההוראה לפי סוג ההוראה. אם נתגלתה שגיאה, יש להודיע עליה.
15. בנה את הקידוד הבינארי של ההוראה, באופן מלא או באופן חלקי ככל שהדבר אפשרי (את מה שלא ניתן לקודד כעת, יש להשלים במעבר השני). הוסף את הקידוד לתמונה הזיכרון של ההוראות. רשום בתמונה ההוראות גם את הערך הנוכחי של IC כתובות הזיכרון של ההוראה נוספת.
16. עדכן $4 \leftarrow IC + 1$, וחזור ל-2.
17. קובץ המקור נקרא בשלמותו. אם נמצאו שגיאות במעבר הראשון, עצור כאן (לא יהיה מעבר שני ולא ייבנו קבצי פלט).
18. שומר את הערכיהם הסופיים של IC ושל DC (נקרא להם ICF ו- DCF). השתמש בהם לבניית קבצי הפלט, אחורי המעבר השני.
19. עדכן בטבלת הסמלים את ערכו של כל סמל המופיע כ-*data*, ע"י הוספה הערך ICF (ראו הסבר לכך בהמשך).
20. עדכן בתמונה הזיכרון של הנתונים את הכתובות של כל הנתונים (כפי שנרשמו בצעד 8), ע"י הוספה הערך ICF לכל כתובה.
21. התחל מעבר שני.

מעבר שני

1. קרא את השורה הבאה בקוד המקור. אם נגמר קוד המקור, עברו ל-9.
2. אם זוהי שורת העלה או שורה ריקה, חזור ל-1.
3. אם השדה הראשון בשורה הוא סמל (תווית), דלג עליו.
4. האם זוהי שורת החלטה שאינה החלטה? אם כן, חזור ל-1.
5. האם זוהי החלטה *entry*? אם לא, עברו ל-7.
6. הוסף בטבלת הסמלים את המאפיין *entry* למאפייני הסמל המופיע כאופרנד של ההחלטה (אם הסמל לא נמצא בטבלת הסמלים, יש להודיע על שגיאה). חזור ל-1.
7. זוהי שורת הוראה. השלים בתמונה הזיכרון את הקידוד הבינארי החסר של ההוראה בעורת בטבלת הסמלים. בהוראה מסוג J (מלבד stop) יש לקודד את כתובת התווית של האופרנד. בהוראת הסתעפות מותנית יש לחשב ולקודד את המרחק מכתובת ההוראה הנוכחית אל כתובות היעד (כתובות התווית של אופרנד היעד). ולהכניס מרחק זה לשדה המתאים בקידוד ההוראה. אם נדרש סמל שאינו נמצא בטבלת הסמלים, או במקרה של הסתעפות מותנית אם נדרש סמל המופיע כ-*external*, יש לדוח על שגיאה ולחזור ל-1.
8. אם בצעד 7 נעשה שימוש בסמל שמופיע כ-*external* (בהוראה מסוג J), הוסף את כתובות ההוראה הנוכחיות, יחד עם שם הסמל, לרשימה הוראות שימושיות בסמל חיצוני (רשימה זו תשמש לבניית קבצי הפלט – ראו בהמשך). חזור ל-1.
9. קוד המקור נסרק בשלמותו. אם נמצאו שגיאות במעבר השני, עצור כאן (לא ייבנו קבצי פלט).
10. בנה את קבצי הפלט (פרטים נוספים בהמשך).

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

להלן שוב תכנית הדוגמה.

```

MAIN: add    $3,$5,$9
LOOP: ori    $9,-5,$2
      la     val1
      jmp   Next
Next: move   $20,$4
      bgt   $4,$2,END
      la     K
      sw    $0,4,$10
      bne   $31,$9,LOOP
      call   val1
      jmp   $4
END: stop
STR: .asciz "aBcd"
LIST: .db    6,-9
        .dh    27056
.entry K
K:    .dw    31,-12
.extern val1

```

בוצע מעבר ראשון על הקוד לעיל, ובננה את טבלת הסמלים. כמו כן, נקודד במעבר זה את כל תמונות הנטוניות, וכן חלקים מהתמונות ההוראות, ככל שהדבר אפשרי בשלב זה.
נסמן "?" בתמונות ההוראות בכל שדה שלא ניתן לקידוד במעבר הראשון.

Address (decimal)	Source Code	Machine Code (binary)
0100	MAIN: add \$3,\$5,\$9	000000 00011 00101 01001 00001 000000
0104	LOOP: ori \$9,-5,\$2	001101 01001 00010 1111111111111011
0108	la val1	011111 0 ?
0112	jmp Next	011110 0 ?
0116	Next: move \$20,\$4	000001 10100 00000 00100 00001 000000
0120	bgt \$4,\$2,END	010010 00100 00010 ?
0124	la K	011111 0 ?
0128	sw \$0,4,\$10	010110 00000 01010 0000000000000100
0132	bne \$31,\$9,LOOP	001111 11111 01001 ?
0136	call val1	100000 0 ?
0140	jmp \$4	011110 1 0000000000000000000000000000100
0144	END: stop	111111 0 00000000000000000000000000000000
0148	STR: .asciz "aBcd"	01100001
0149		01000010
0150		01100011
0151		01100100
0152		00000000
0153	LIST: .db 6,-9	00000110
0154		11110111
0155	.dh 27056	0110100110110000
0157	K: .dw 31,-12	0000000000000000000000000000000011111
0161		111111111111111111111111111111110100

טבלת הסמלים אחרי מעבר ראשון היא:

Symbol	Value (decimal)	Attributes
MAIN	100	code
LOOP	104	code
Next	116	code
END	144	code
STR	148	data
LIST	153	data
K	157	data
val1	0	external

נבע עתה את המעבר השני. נשלים באמצעות טבלת הסמלים את הקידוד במקומות המסומנים ב- "י?".
הקוד הבינארי בקורסו הסופית כאן זהה לקוד שהוגן בתחלת הנושא "אסטמבלר עם שני מעברים".

טבלת הסמלים אחרי מעבר שני היא :

Symbol	Value (decimal)	Attributes
MAIN	100	code
LOOP	104	code
Next	116	code
END	144	code
STR	148	data
LIST	153	data
K	157	data, entry
val1	0	external

בסוף המעבר השני, אם לא נתגלו שגיאות, האסמבלר בונה את קבצי הפלט (ראו בהמשך), שמכילים את הקוד הבינארי ומידע נוסף עבור שלב הקישור.

הערה : כאמור, האסמבלר בונה קוד מכונה כך שיתאים לטעינה לזכרון החל מכתובת 100 (עשוריוני). אם הטעינה בפועל (לצורך הרצת התוכנית) תהיה לכתובות אחרות, יידרש תיקונים בקידוד של חלק מהחראות מסווג J בעט הטעינה.

כאמור, שלבי הקישור והטעינה איןם למימוש בפרויקט זה, ולא נדוע בהם כאן.

קבצי קלט והפעלת האסמבלר

בפעולת האסמבלר, יש להעביר אליו באמצעות ארגומנטים של שורת הפקודה (command line arguments) רשימה של שמות של קבצי קלט (אחד או יותר). אלו הם קבצי טקסט, ובהם תוכניות בתחריר של שפת האסמבלי שהוגדרה במינ' זה.

שם של קובץ המכיל תכנית בשפת אסמבלי חייב להיות עם הסיומת ".as". למשל, hello.as הוא שם תקין.

לדוגמה : נניח שתוכנית האסמבלר שלו נקראת assembler, אז שורת הפקודה הבאה :

```
assembler test.as myprog.as hello.as
```

תירץ את האסמבלר על שלשת קבצי הקלט שהם הארגומנטים בשורת הפקודה.

האסמבלר פועל בנפרד על כל קובץ קלט בראשית הארגומנטים, אחד אחרי השני. אם קובץ קלט לא נפתח, יש להדפיס הودעת שגיאה ולבור לקובץ הבא. אם קובץ הקלט עבר את תהליך האסמבלי ללא שגיאות, האסמבLER בונה עבورو קבצי פلت.

קבצי פلت של האסמבLER

עבור כל קובץ קלט שעבר אסמבלי ללא שגיאות, האסמבLER יוצר קבצי פلت כלהלן.

- קובץ object, המכיל את קוד המcona.
- קובץ externals, ובו פרטימ על כל המKENOTOT (הכתובות) בקוד המcona בהם מקודד סמל שהוצחר חייצוני (סמל שהופיע כאופרנד של הנחיתת extern., ומופיע בביטול הסמלים - external.).
- קובץ entries, ובו פרטימ על כל סמל שמוצהר כנקודות כניסה (סמל שהופיע כאופרנד של הנחיתת entry., ומופיע בביטול הסמלים - entry).

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

שמות קבצי הפלט מבוססים על שם קובץ הקלט, כפי שהופיע בשורת הפקודה, בתוספת סימנת מתאימה: הסימנת `"ob."` עבור קובץ ה-`object`, `"entries"` עבור קובץ ה-`entries`, והסימנת `"ext"` עבור קובץ ה-`externals`.

לדוגמה, בהפעלת האסטמבלר באמצעות שורת הפקודה `assembler myprog.as` יוצר קובץ פلت `myprog.ob`, וכן קבצי פلت `myprog.ext` ו- `myprog.entries` ככל שיש הנחיתות `extern` או `entry`. או בקובץ המקור.

ນציג CUT את הפורמטים של קבצי הפלט. דוגמאות יובאו בהמשך.

פורמט קובץ ה- `object`

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

כזכור, האסטמבלר מ庫ודד את ההוראות כך שתמונה ההוראות תתאים לטעינה החל מctaובת 100 (עשרוני) בזיכרונו. נשים לב שרף המעבר הראשון יודיעים מהו הגודל הכללי של תמונה ההוראות. מכיוון שתמונה הנתוניות נמצאת אחרי תמונה ההוראות, גודל תמונה ההוראות משפיע על הכתובות בתמונה הנתוניות. זו הסיבה שבגללה היה צריך לעדכן בטבלת הסמלים, בסוף המעבר הראשון, את ערכי הסמלים המאופיינים כ-`data` (כזכור, באלווריותם השלדי שהוצג לעיל, בצד 19, הוספנו לכל סמל כזה את הערך ICF). במעבר השני, בהשלמת הקידוד, משתמשים בערכים המעודכנים של הסמלים, המותאמים למבנה המלא והסופי של תמונה הזיכרונו.

CUT האסטמבלר יכול לכתוב את תמונה הזיכרונו בשלמותה לתוכן קובץ פلت (`object`). הקובץ בניוי משורות טקסט בפורמט שלහן.

השורה הראשונה בקובץ ה- `object` היא `"content"`, המכילה שני מספרים בסיסי עשרוני: הראשון הוא האורך הכולל של תמונה ההוראות (כלומר כמה תאי הזיכרונו), והשני הוא האורך הכולל של תמונה הנתוניות (כמה תאי הזיכרונו). בין שני המספרים מפריד רווח אחד. כזכור, במעבר הראשון, בצד 18 באלווריותם, נשמרו הערכים IDF ו- ICF. האורך הכולל של תמונה ההוראות הוא 100-ICF, והאורך הכולל של תמונה הנתוניות הוא IDF.

השורות הבאות בקובץ מכילות את תמונה הזיכרונו, לפי סדר עולה של כתובות. בכל שורה 5 שדות: כתובות בזיכרונו ולאחריה התכנים של 4 בתים שמנצאים החל מctaובת זו, בסדר עולה משמאלי לימינו בשורה. הכתובות תירשם בסיס עשרוני בארכיטקטורה little-endian (כוללAPS מוביילים). התוכן של כל בית יירשם בסיס הקסאדיימלי בשתי ספרות (כוללAPS מובייל), כאשר הספרות F-A באותיות גדולות. בין שדות בשורה מפריד רווח אחד.
השורה الأخيرة בקובץ יכולה להכיל פחוות מ- 4 בתים, לפי הצורך.

ההוראות מוכנה מקודדות ל-4 בתים, ולפיכך כל הוראה תפוסף שורה שלמה בקובץ ה-`object`.
כנאמר, הזיכרונו מאורגן בשיטת little-endian, ולכן סיביות 7-0 של ההוראה יהיו הבית השמאלי בסדר הבתים בשורה, סיביות 15-8 הבית השני משמאלי בשורה, וכן הלאה.

הנתוניות, לעומת זאת, מקודדים באורכים שאינם בהכרח כפולה של 4 בתים. למרות זאת, יש להכניס את הנתוניות לקובץ ה-`object` ברביעיות של בתים בכל שורה, ללא "חורים", ותוך הקפדה על הסדר הנכון (סדר עולה של כתובות בכל שורה משמאלי לימין, וסדר עולה משורה לשורה). ראו דוגמה בהמשך.

פורמט קובץ ה-**entries**

קובץ ה-**entries** בנייתן מושוראות טקסט, שורה אחת לכל סמל שמאופיין בטבלת הסמלים כ-**entry**. בשורה מופיע שם הסמל, ולאחריו ערכו כפי שנקבע בטבלת הסמלים (בבסיס עשרוני באربע ספרות, כולל אפסים מוביילים). בין שני השדות בשורה יש רווח אחד. אין חשיבות לסדר השורות, כי כל שורה עומדת בפני עצמה.

פורמט קובץ ה-**externals**

קובץ ה-**externals** בנייתן מושוראות טקסט, שורה לכל הוראה מסווג **J** בקוד המconaה בה יש שימוש בסמל שמאופיין כ-**external**. כזכור, רשיימה של הוראות אלה נבנתה בעבר השני (צעד 8 באלגוריתם השודי).

כל שורה בקובץ ה-**externals** מכילה את שם הסמל החיצוני, ולאחריו כתובות ההוראה בה הוא נדרש (בבסיס עשרוני באربע ספרות, כולל אפסים מוביילים). בין שני השדות בשורה יש רווח אחד. אין חשיבות לסדר השורות, כי כל שורה עומדת בפני עצמה.

لتשומת לב: ניתן ויש מספר כתובות בקוד המconaה בהן נדרש כתובתו של הסמל החיצוני, ככל כתובות צוות תהיה שורה נפרדת בקובץ ה-**externals**.

נדגים את הפלט שמייצר האסמבילר עבור קובץ מקור בשם ps.as הנתון להלן.

```
;file ps.as
;sample source code

.entry Next
.extern wNumber
STR: .asciz "aBcd"
MAIN: add $3,$5,$9
LOOP: ori $9,-5,$2
      la val1
      jmp Next
Next: move $20,$4
LIST: .db 6,-9
      bgt $4,$2,END
      la K
      sw $0,4,$10
      bne $31,$9, LOOP
      call val1
      jmp $4
      la wNumber
.extern val1
      .dh 27056
K:   .dw 31,-12
END  stop
.entry K
```

להלן הקידוד הבינארי המלא (תמונת הזיכרון) של קובץ המקור, בגמר מעבר השני.

טבלת הסמלים בגמר המעבר השני היא:

Symbol	Value (decimal)	Attributes
wNumber	0	external
STR	152	data
MAIN	100	code
LOOP	104	code
Next	116	code, entry
LIST	157	data
val1	0	external
K	161	data, entry
END	148	code

להלן תוכן קבצי הפלט של הדוגמה.

:ps.ob הקובץ

```
52 17
0100 40 48 65 00
0104 FB FF 22 35
0108 00 00 00 7C
0112 74 00 00 78
0116 40 20 80 06
0120 1C 00 82 48
0124 A1 00 00 7C
0128 04 00 0A 58
0132 E4 FF E9 3F
0136 00 00 00 80
0140 04 00 00 7A
0144 00 00 00 7C
0148 00 00 00 FC
0152 61 42 63 64
0156 00 06 F7 B0
0160 69 1F 00 00
0164 00 F4 FF FF
0168 FF
```

:ps.ent הקובץ

Next 116
K 0161

:ps.ext הקובץ

```
val1 0108
val1 0136
wNumber 0144
```

סיכום והנחיות כלליות

- גודל תכנית המקור הניתנת כקלט לאסטמבלר אינו ידוע מראש, ולכן גם גודלו של קוד המוכנה אינו צפוי מראש. אולם בכך להקל בימוש האסטמבלר, מותר להניח גודל מקסימלי. לפיכך יש אפשרות להשתמש במערכות לאחסן תמונות קוד המוכנה בלבד. כל מבנה נתונים אחר (למשל טבלת הסמלים), יש למשם באופן ייעיל וחסכוני (למשל באמצעות רישימה מקוישת והקצתה זיכרון דינמי).
- השימוש של קבצי הפלט צריים להיות תואמים לשם קובץ הקלט, למעט הסיומות. למשל, אם קובץ הקלט הוא prog.as אז קבצי הפלט שיוצאו הם : prog.ob, prog.ext, prog.ent.
- מתוכנות הפעלת האסטמבלר צריכה להיות כפי הנדרש בממ"ן, ללא שינוים כלשהם. ככלומר, ממשך המשמש יהיה אך ורק באמצעות שורת הפוקודה. בפרט, שמות קבצי המקור יעברו לתוכנית האסטמבלר כארגומנטים (אחד או יותר) בשורת הפוקודה. אין להוסיף תפריטי קלט אינטראקטיביים, חלונות וגרפיים למיניהם, ועוד'.
- יש להקפיד לחלק את שימוש האסטמבלר למספר מודולים (קבצים בשפת C) לפי מישימות. אין לרכז מישימות מסוימים במודול יחיד. מומלץ לחלק למודולים כגון : מעבר ראשון, מעבר שני, פונקציות עזר (למשל, תרגום לביסיס, ניתוח תחבירי של שורה), טבלת הסמלים, מפת הזיכרון, טבלאות קבועות (כגון קוד הפעולה)
- יש להקפיד ולתעד את השימוש באופן מלא וברור, באמצעות הערות מפורחות בקוד.
- יש לאפשר תווים לבנים עודפים בקובץ הקלט בשפת אסטמבלר. למשל, אם בשורת הוראה יש שני אופרנדים המופרדים בפסיק, אין לפני ואחריו הפסיק מותר שיהיו רווחים וטאבים בכל 경우에는. בדומה, גם לפני ואחרי שם הפעולה. מותר גם שורות ריקות. האסטמבלר יתעלם מהתווים לבנים מיוטרים (כלומר ידלג עליהם).
- הקלט (קוד האסטמבלר) עלול להכיל שגיאות תחביריות. על האסטמבלר �גלוות ולדוח על כל השורות השגויות בקלט. אין לעצור את הטיפול בקובץ קלט לאחר גילוי השגיאה הראשונה. יש להדפיס למסך הודעות מפורטות לכל הניתן, כדי שאפשר יהיה להבין מה והיכן כל שגיאה. כמובן שגם קובץ קלט מכיל שגיאות, אין טעם להפיק עבورو את קבצי הפלט (ob, ext, ent).

תמ ונסלם פרק ההסברים והגדרת הפרויקט.

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

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

לתשומתיכם : לא תינטו דחיה בהגשת הממי"ן, פרט למקרים מיוחדים כגון מיליאים או מחלוקת ממושכת. במקרים אלו יש לבקש ולקבל אישור מראש ממנהל הקבוצה.

בזה לחה !