

האוניברסיטה הפתוחה

20465

מעבדה בתכנות מערכות

חוברת הקורס - סתיו 2010א

כתבה: מיכל אבימור

אוקטובר 2009 – סמסטר סתיו – תש"ע

פנימי – לא להפצה.

© כל הזכויות שמורות לאוניברסיטה הפתוחה.

תוכן העניינים

א	אל הסטודנט
	מתכונת הקורס
ה	1. תיאור הקורס
ה	2. כיצד ללמוד
ו	3. מפגשים
ז	4. התרגול בקורס
ז	5. סטודנטים בעלי מחשב אישי
ח	6. בחינות הגמר
ח	7. התנאים לקבלת נקודות זכות בקורס
ט	8. למידה מתוקשבת ואתר הקורס באינטרנט
יב	9. לוח זמנים ופעילויות
	מטלות הקורס
יז	10. תיאור המטלות
יח	11. הנחיות לכתיבת מטלות וניקודן
כ	12. נוהל הגשת מטלות
1	ממ"ן 11
3	ממ"ן 12
9	ממ"ן 13
11	ממ"ן 14

אל הסטודנט,

אני מקדמת את פניך בברכה, עם הצטרפותך אל הלומדים בקורס "מעבדה בתכנות מערכות". בחוברת זו ניתן למצוא תיאור, מלא ככל האפשר, של הקורס וכן פרטים על כלל הפעילויות, במהלך הלימודים. רצוי לראות בה כעין מדריך אישי, שתפקידו להבהיר עניינים שונים. יש לקרוא בעיון רב את כל הסעיפים. בהמשך ניתן למצוא את לוח הזמנים ואת מטלות הקורס.

פרטים נוספים על המערכת המסייעת ללימוד עצמי, מרכיביה ופרטים מינהליים לביצוע הפעילויות השונות במסגרת לימודך, ניתן למצוא בקטלוג הקורסים ובידיעון האקדמי. עדכונים יישלחו מדי סמסטר.

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

אני מאחלת לך לימוד פורה ומהנה.

בברכה,

מיכל אבימור
מרכזת ההוראה בקורס.

מתכונת הקורס

1. תיאור הקורס

הקורס מבוסס בין היתר על הספר:

“The C Programming Language”

שנכתב על ידי:

Brian W. Kernighan ו-Dennis M. Ritchie.

לספר מצורף מדריך למידה, שתפקידו להנחות את הסטודנט בלימוד הקורס. משימות הלימוד לכל שבוע, והתאריך האחרון למשלוח כל אחת ממטלות הקורס רשומים ב"לוח זמנים ופעילויות".

חומר הלימוד כולל את פרקים 1-8 בספר דלעיל ואת החומר המופיע במדריך הלימוד.

2. כיצד ללמוד

הקורס מבוסס על פרקים 1 – 8 בספר הלימוד, בתוספת מדריך למידה, נספחי התוכניות לדוגמא, והמטלות. מדריך הלימוד מכיל הנחיות לגבי עיקרי הדברים אותם יש להבין ולדעת. כמו כן מכיל המדריך דוגמאות נוספות, שאלות ותשובות. המדריך מהווה את נקודת המוצא לתהליך הלימודי ומלווה את שמונת הפרקים בספר הקורס.

השאלות המופיעות במדריך הלימוד נועדו להכשיר אותך לענות על שאלות מורכבות יותר. רצוי לענות ו"להריץ" כל שאלה המופיעה במדריך, ואחר כך להשוות את התשובה שלך לזו המופיעה במדריך הלימוד.

בנוסף, לרשותך לקט של תוכניות לדוגמא, המדגישות את הייחוד של שפת C. לקט זה מופיע כנספח במדריך הלימוד.

במקרה של קושי תוך כדי לימוד, רצוי לנצל את אתר הקורס ואת ההנחיה הטלפונית או לשאול את שאלתך במפגש עם המנחה. לתשומת לבך, ניתן לפנות למנחה של הקבוצה שלך או לכל מנחה אחר, המנחה במסגרת הקורס. בתחילת הסמסטר תשלח אליך רשימה של המנחים בקורס, שעות ההנחיה הטלפונית ומספרי הטלפון שלהם.

לאחר שנראה לך שהבנת היטב את חומר הלימוד, יש לגשת לפתרון המטלה. המטלה כוללת, בדרך כלל, שאלות בדומה לאלה המופיעות בפרקי הלימוד, והן נועדו לבדוק את יכולתך ביישום חומר הלימוד.

לימוד שיטתי של פרקי הלימוד, יחד עם פתרון המטלות, יקנה לך הכנה מלאה לקראת בחינת הגמר. הבחינה עשויה לכלול שאלות שבהן נדרשים לתאר ולהבין תוכניות קיימות, מתן הסברים על פעולות שונות מהשפה, וכן שאלות הדורשות כתיבת תוכנית, פונקציה או קטע קוד.

שמירה על קצב הלימוד המומלץ , והגשת המטלות בזמן, ימנעו ממך קשיים בלתי רצויים במהלך מחזור הלימודים, ויסייעו לך בהפקת מלוא התועלת מהקורס.

3. מפגשים

במהלך הסמסטר יתקיימו 7 מפגשי הנחיה במרכז הלימוד. מפגשי ההנחיה יארכו כשעתיים וחצי כל אחד, ויהיו מפגשים עיוניים, שיתקיימו בחדר לימוד. מפגשים אלה נועדו להבהיר את החומר הנלמד עד למועד המפגש, ולעזור לך להתגבר על קשיים, בהבנה או בפתרון של שאלות בחומר הלימוד ובמטלות. בכל מפגש יוקדש חלק מן הזמן להבהרת נקודות מרכזיות מהחומר שביחידת הלימוד השוטפת, ועיקר הזמן הנותר יוקדש לשאלות הסטודנטים ולדיון במטלה. כמו כן ייתן המנחה רקע להכנת המטלה הבאה, שעליך להגיש ויכוון אותך אל הגישה הנכונה לפתרונה.

שים לב!

ההשתתפות במפגש ההנחיה אינה חובה אך היא בהחלט רצויה!

4. התרגול בקורס

מרכזי הלימוד, ובהם המחשבים, יעמדו לרשותך לצורך כתיבת התכניות והרצתן. לבירור המועדים ושעות התרגול של קבוצתך עיין ב"לוח מפגשים ומנחים".

התרגול יתבצע בעזרת מערכת ההפעלה Linux. המערכת והמהדר (compiler) יותקנו במרכזי הלימוד לשימוש הסטודנטים. חוברת הדרכה ומדריך לשימוש במערכת זו, מצורפת כנספח למדריך הלמידה.

5. סטודנטים בעלי מחשב אישי

התרגול בקורס יתבצע על גבי מחשבים אישיים (PC). אם ברשותך מחשב אישי מתאים, תוכל להשתמש בו לכתיבת התוכניות בקורס ולהרצתן. לצורך כתיבת תכניות במחשבים בביתכם, תזדקקו לערכה להתקנת מערכת ההפעלה Linux, המצורפת לחומר הלימוד. דרישות החומרה והתוכנה, הנחוצות להרצת המערכת והמהדר, מופיעות אף הן במדריך ההתקנה למערכת Linux, המצורף לחומר הלימוד, וכן בנספח במדריך הלמידה.

6. בחינות הגמר

הנך זכאי לגשת לבחינת גמר בקורס רק אם עמדת בכל דרישות הקורס לפני מועד בחינה. (כלומר הגשת מטלות במשקל מינימלי והשתתפת בשאר פעילויות החובה של הקורס).

בחינות הגמר יחלו כשבוע ימים לאחר תום הסמסטר. הודעה על המועדים המדויקים תישלח לסטודנטים על-ידי מרכז ההישגים הלימודיים כחודשיים לאחר תחילת הסמסטר. מועדי בחינות הגמר שנקבעו לסמסטרים הבאים מפורטים בידיעון האקדמי.

לתשומת לב!

הנך זכאי להיבחן בקורס פעמיים: במועדים של הסמסטר הנוכחי או במועדים של הסמסטר הבא בו נלמד הקורס, ובכך מיצית את זכותך להיבחן בקורס. סטודנט שניגש לבחינות גמר בשני מועדים ונכשל בשניהם, יוכל להירשם לקורס זה פעם נוספת ולקבל הנחה בשכר הלימוד. פרטים בידיעון האקדמי.

ספר הקורס מותר לשימוש בזמן הבחינה.

חל איסור מוחלט על שימוש במחשב מכל סוג (כולל מחשב כיס, מחשב נישא וכו') בזמן הבחינה.

7. התנאים לקבלת נקודות זכות בקורס

- א. חובה להגיש שלוש מטלות בקורס (כולל פרוייקט הגמר).
- ב. ציון של לפחות 60 נקודות בבחינת הגמר.
- ג. ציון סופי בקורס של 60 נקודות לפחות.

8. למידה מתוקשבת ואתר הקורס באינטרנט <http://telem.openu.ac.il>

לקורס שבו אתם לומדים קיים אתר באינטרנט הפועל כמעין מרכז לימוד וירטואלי של הקורס. האתר מהווה עבורכם ערוץ תקשורת עם סטודנטים אחרים בקורס ועם צוות ההוראה, ומאפשר לכם ליהנות מחומרי למידה נוספים שמפרסם מרכז ההוראה. ההשתתפות בפעילות המתוקשבת באתר אינה דורשת הרשמה מיוחדת. הכניסה לאתר מתבצעת מכל עמדת מחשב שיש בה חיבור לאינטרנט (בבית, במקום העבודה, ממחשב של חבר), בשעות ובימים הנוחים לכם.

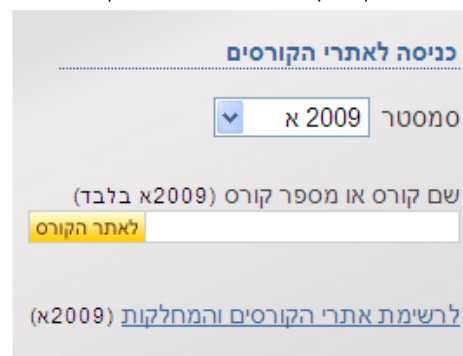


מהם הציוד והתוכנה הנדרשים כדי לגלוש באתר?

כדי לבקר באתר ולהשתתף בפעילות נדרשת גישה למחשב המסוגל להריץ Microsoft Internet Explorer 6 ומעלה, הכולל מעבד התמלילים Microsoft Word 7.0 ומעלה. תוכנות Office אחרות מומלצות.

כיצד מגיעים לאתר הקורס?

תחילה עליכם להיכנס לאתר הראשי של שוהם בכתובת: <http://telem.openu.ac.il>
לאחר מכן הקלידו את מספר הקורס או את שמו בחלון שלהלן:



The screenshot shows a web page titled "כניסה לאתרי הקורסים" (Login to course sites). It features a dropdown menu set to "סמסטר 2009 א" (Semester 2009 A). Below this is a text input field labeled "שם קורס או מספר קורס (2009 א בלבד)" (Course name or course number (2009 A only)). A yellow button labeled "לאתר הקורס" (Find the course) is next to the input field. At the bottom, there is a link: "לרשימת אתרי הקורסים והמחלקות (2009 א)" (To the list of course sites and departments (2009 A)).

מה כוללים אתרי הקורסים?

אתרי הקורסים מאפשרים לקיים **תקשורת זמינה ושוטפת** בין כל השותפים ללמידה ולהוראה בקורס.

נוסף על כך באתרי הקורסים מתפרסמים **חומרי לימוד** כגון: עדכונים ליחידות הלימוד, תרגול נוסף, דוגמאות של מבחנים, משובים לממ"נים, המחשות, לומדות ועוד. **חומרי העשרה** כגון: מצגות, עבודות לדוגמה של סטודנטים, נושאים אקטואליים, מבחני רב ברירה עם משוב מיידי, קישורים למאגרי מידע ולאתרים שונים ברשת האינטרנט ועוד.

בחלק מהאתרים משולבים **שיעורי וידיאו** מוקלטים המחולקים לפרקים והמזמנים לימוד הדומה במקצת לשיעור חי. החלוקה לפרקים מאפשרת צפייה נוחה בשיעור, ובמיוחד חזרה על פרקים ספציפיים מתוך הרצף. בדקו האם יש הפניה לשיעורי וידיאו בקורס שלכם והיעזרו בהם ללמידה. כל אלה הן דוגמאות בלבד - באתר של כל קורס בוחר מרכז ההוראה להציג את החומרים המתאימים לתכני הקורס.


הפנקס האישי

באתרי הקורסים משולב "פנקס אישי" המאפשר לכם לרכז הערות אישיות לחומרים שתבחרו מתוך אתר הקורס. הפנקס האישי, כשמו כן הוא - אישי. רק אתם מורשים לצפות בו. אותו פנקס ילווה אתכם בכל תקופת לימודיכם באוניברסיטה הפתוחה וישרת אתכם בכל הקורסים שתלמדו. תוכלו לאסוף לפנקס האישי פריטי תוכן מאתרי קורסים שונים, בתנאי שיש לכם הרשאה אליהם. פרטים על הפנקס האישי והמלצות לשימוש בו ראו באתר תלם, אזור מידע לסטודנטים או ישירות בכתובת: http://telem.openu.ac.il/personal_notes מקווים שהפנקס האישי יהיה לכם לעזר במהלך לימודיכם באוניברסיטה הפתוחה.

כיצד מתבצעת התקשורת באתר?

בדף הבית באתר פרוס לוח הודעות בו מתפרסמות הודעות שוטפות מטעם צוות ההוראה בנושאים ואירועים הקשורים לקורס. באתר יש קבוצת דיון המאפשרת שיח שוטף בין כל משתתפי הקורס באמצעות חילופי טקסט. אפשר לשתף ולהתייעץ, לדון בחומר הלימוד, להעלות קשיים, לשאול שאלות ולקיים שיח לימודי וחברתי. קבוצת הדיון פתוחה רק בפני הסטודנטים והמנחים הלומדים ומלמדים בקורס. הדואר האלקטרוני מאפשר קיום תקשורת בינאישית בין הסטודנטים ומול צוות ההוראה. הציט מאפשר לכל משתתפי הקורס, לומדים ומלמדים, "לשוחח" בזמן אמת באמצעות הודעות טקסט במועד שנקבע מראש.

ביקור ראשון באתר הקורס

הצעד הראשון בביקורכם באתר הוא לערוך עימו הכרות - התחילו לשוטט במדורים השונים הנמצאים באתר בצורה חופשית כדי להכיר את המבנה שלו ואת התכנים שנמצאים בו. היכנסו ל  עדכון פרטים אישיים ובצעו את הפעולות הבאות:

- **עדכן את כתובת הדואר האלקטרוני שלכם** כדי שתוכלו לקבל דואר ממרכז ההוראה.
- **אשרו פרסום שמכם** בדף רשימות הסטודנטים באתר כדי שסטודנטים אחרים יוכלו לפנות אליכם ישירות.
- **תוכלו לשנות את סיסמת הגישה האישית לאתר** (אם היא מסובכת מדי לזכירה). בקרו בקבוצת הדיון והציגו עצמכם בפני צוות הקורס וחברי הקבוצה, תוכלו לספר מעט על עצמכם ולשתף אחרים בציפיות שלכם מהקורס. בביקורים הבאים באתר, נצלו את קבוצת הדיון להעלות שאלות, להציע רעיונות ולשתף אחרים בחוויות ובפתרונות. לרשותכם קיים באתר מדריך למשתמש הכולל הנחיות טכניות לתפעול סביבת הלמידה, אליו ניתן להגיע מהקישור [עזרה](#) בראש דף הבית.

תדירות הביקור באתר ולמה כדאי לחזור ולבקר בו

האינטרנט כידוע הוא מדיום בעל יתרונות רבים, אחד מהם הוא האפשרות לעדכן את המידע באופן שוטף ובמהירות. היתרון הזה בא לידי ביטוי באתרי הקורסים ומאפשר לצוות ההוראה לעדכן את האתר ואתכם, הסטודנטים, באופן שוטף בפרסומים, בחידושים, בדוגמאות אקטואליות ועוד. במילים אחרות, בניגוד ליחידות הלימוד הכתובות, אתר הקורס כפי שמוצג בראשית הסמסטר אינו דומה כלל וכלל לאתר הקורס בסוף הסמסטר. אתרי הקורסים מתרחבים ומתעדכנים כל העת. עשו לעצמכם מנהג לבקר באתר באופן שגרתי ולהפנות אליו את שאלותיכם. גם אם בהתחלה הדבר יהיה אולי מכביד או מאולץ, עם הזמן תיווכחו כי עומד לרשותכם אמצעי עזר יעיל ללמידה.

היכנסו לאתר, היעזרו בתכנים השונים וכמובן השתתפו באופן פעיל. האתר נועד לכם ושימוש נכון בו יכול להקל עליכם את הלמידה.
להתראות באתר!

כיצד מקבלים סיסמת גישה לאתר הקורס?

לכל סטודנט הרשום לקורס מתוקשב, נפתח באוניברסיטה חשבון אישי הכולל סיסמת גישה לאתר הקורס באינטרנט. הסיסמה מופקת פעם אחת לכל תקופת הלימודים, ותשרת אתכם בכל הקורסים המתוקשבים שאליהם אתם רשומים. **חשוב לשמור את הסיסמה גם לקורסים ולסמסטרים הבאים.** אם זו פעם ראשונה שאתם לומדים בקורס מתוקשב, תישלח לביתכם הודעה שתכלול את שם המשתמש והסיסמה המקורית שלכם. **אנא הקפידו לשמור פרטים אלה!** תוכלו לשנות את הסיסמה האישית באתר הקורס בכפתור **עדכון פרטים אישיים**. אם שנייתם את הסיסמה, אנא הקפידו לרשום אותה לפניכם. אם שכחתם אותה, עליכם ליצור קשר עם מוקד הפניות והמידע בטלפון 09-7782222, באמצעות דואר אלקטרוני: infodesk@openu.ac.il או תוכלו להשתמש גם בשירותי קול האו"פ בטלפון 09-7781111.

שימו לב! מטעמי סודיות לא ניתן לקבל את הסיסמה בטלפון. בכל מקרה של דרישת סיסמה, היא תישלח בדואר לכתובת המעודכנת במחשב האוניברסיטה הפתוחה.

שליחת ממ"נים באמצעות מערכת המטלות המקוונת

בכל קורס (למעט בודדים), ניתן להגיש מטלות באמצעות מערכת המטלות המקוונת. מערכת המטלות המקוונת היא, מערכת ממוחשבת מבוססת אינטרנט לשינוע מטלות מן הסטודנטים למנחים ובחזרה. המטלות נשלחות באמצעותה מהסטודנטים למנחי הקורס ומוחזרות לאחר בדיקתן כולל ציון ומשוב, תוך בקרה מלאה של מרכזי ההוראה. יתרונותיה הבולטים של המערכת, היא האפשרות של הסטודנטים לדעת בכל שלב האם המטלה נמצאת אצל המנחה (הורדה למחשב שלו), האם נבדקה, ומה הציון שניתן עליה. על כל אלה יש להוסיף את היתרון כי שימוש במערכת המקוונת אינו מצריך מילוי ידני של טפסים וכמובן שאין צורך במשלוח בדואר. לצד המעקב המנהלי, המערכת מאפשרת, קבלת משוב מסודר ומתועד היטב בגוף המטלה או בקובץ נפרד.

תמיכה טכנית ובירורים



מוקד הפניות והמידע

טלפון רב קווי 09-7782222, דואר אלקטרוני: infodesk@openu.ac.il
שעות הפעילות של מוקד הפניות הן:

בימי ראשון עד חמישי בין השעות: 8:30 - 19:00

בימי שישי וערבי חג בין השעות: 8:30 - 12:30

בעת הפנייה למוקד, הנכם מתבקשים להצטייד במספר ת"ז וקוד אישי.

יש לפנות למוקד בנושאים:

- סיסמת המשתמש (לקבלה או שחזור סיסמה. ניתן גם להשתמש גם בשירותי קול האו"פ בטלפון 09-7781111)
- הודעת שגיאה המודיעה כי אינכם מורשים לגשת לדף כלשהו באתר
- קשיים בהפעלת מערכת שליחת מטלות (במידה שקיבלתם הודעה שבקורס נעשה שימוש במערכת)
- שאלות כלליות על אתרי הקורסים ודיווח על תקלות טכניות באתר (למשל דף משובש או כתובת URL שגויה)

בכל הנושאים הקשורים לתכנים באתר הקורס, עליכם לפנות לצוות ההוראה בקורס.

9. לוח זמנים ופעילויות (2010א/20465)

שבוע הלימוד	תאריכי שבוע הלימוד	יחידת הלימוד המומלצת	מפגשי ההנחיה*	תאריך אחרון למשלוח הממ"ן (למנחה)
1	23.10.2009-18.10.2009	ספר C פרקים 1-2-3	מפגש ראשון	
2	30.10.2009-25.10.2009	ספר C פרקים 1-2-3		
3	6.11.2009-1.11.2009	ספר C פרק 4	מפגש שני	
4	13.11.2009-8.11.2009	ספר C פרק 4		
5	20.11.2009-15.11.2009	ספר C פרק 5	מפגש שלישי	ממ"ן 11 15.11.2009
6	27.11.2009-22.11.2009	ספר C פרק 5		
7	4.12.2009-29.11.2009	ספר C פרק 6	מפגש רביעי	
8	11.12.2009-6.12.2009	ספר C פרק 6		

* התאריכים המדויקים של המפגשים הקבוצתיים מופיעים ב"לוח מפגשים ומנחים". אנא שבצו אותם בכתב ידכם. מרכז הלימוד ומספר הקבוצה מצוינים בהודעה ללומד שקיבלתם ממערך שירותי הוראה.

לוח זמנים ופעילויות - המשך

שבוע הלימוד	תאריכי שבוע הלימוד	יחידת הלימוד המומלצת	מפגשי ההנחיה*	תאריך אחרון למשלוח הממ"ן (למנחה)
9	18.12.2009-13.12.2009 (א-ו חנוכה)	ספר C פרק 6		
10	25.12.2009-20.12.2009	ספר C פרק 7	מפגש חמישי	ממ"ן 12 20.12.2009
11	1.1.2010-27.12.2009	ספר C פרק 7		
12	8.1.2010-3.1.2010	ספר C פרק 8 + פרויקט	מפגש שישי	
13	15.1.2010-10.1.2010	פרויקט וחזרה		
14	22.1.2010-17.1.2010	פרויקט וחזרה	מפגש שביעי	ממ"ן 13 17.1.2010
15	29.1.2010-24.1.2010	פרויקט וחזרה		ממ"ן 14** 28.3.2010

מועדי בחינות הגמר יפורסמו בנפרד

* התאריכים המדויקים של המפגשים הקבוצתיים מופיעים ב"לוח מפגשים ומנחים". אנא שבצו אותם בכתב ידכם. מרכז הלימוד ומספר הקבוצה מצוינים בהודעה ללומד שקיבלתם ממערך שירותי הוראה.

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

מטלות הקורס

10. תיאור המטלות

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

בקורס זה שלוש **מטלות חובה**, שהן בעיקרן תוכניות מחשב. להלן מספרי המטלות ומשקליהן:

ממ"ן	משקל	פרקים
11	4 (ממ"ן חובה)	3,2,1
12	10 ממ"ן רשות	5,4
13	15 (ממ"ן חובה)	7,6
14	31 (ממ"ן חובה)	פרוייקט גמר

עליך להגיש במהלך הקורס שלוש מטלות מתוך הארבע. את התשובות לממ"נים יש להגיש באמצעות מערכת המטלות (ניתן, אך לא מומלץ, להגיש את המטלות באמצעות הדואר או הגשה ישירה למנחה במפגשי ההנחיה. במקרה כזה יש לתאם את הדבר עם הבודק). יש להגיש את קבצי המקור (.C), קבצי ההרצה, קבצי הסביבה המתאימים (כולל קבצי MAKEFILE), קבצי קלט וקבצי פלט.

לתשומת לבכם!

כדי לעודדכם להגיש לבדיקה מספר רב של מטלות הנהגנו את ההקלה שלהלן: אם הגשתם מטלות מעל למשקל המינימלי הנדרש בקורס, **המטלה** בציון הנמוך ביותר, שציונה נמוך מציון הבחינה, לא תילקח בחשבון בעת שקלול הציון הסופי. זאת בתנאי שמטלה זו **אינה חלק מדרישות החובה בקורס** ושמשקל המטלות האחרות שהוגשו עובר את המינימום ההכרחי.

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

11. הנחיות לכתיבת מטלות וניקודן

ניקוד המטלות ייעשה 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.
- לכל משתנה יהיה שם משמעותי ויוצמד אליו תיעוד לגבי תפקודו בתכנית. i,j,k - משמשים בד"כ כשמות אינדקסים ואין צורך לתעד אותם.
- 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.
- יש להשתמש בשמות משמעותיים ל: פונקציות, מקרואים, משתנים, קבועים, הגדרת טיפוסים וקבצים.
- יש להקפיד על קריאות ובהירות תוך שימוש באינדנטציה (היסח) מסודרת ואחידה.

ג. תכנות - 40%

יש להקפיד על כתיבה מסודרת ומודולרית של קוד :

- חלוקה לקבצים - כשלכל קובץ מוצמד קובץ header אם צריך (כאשר נדרש בתרגיל).
- חלוקה לפונקציות.
- שימוש במקרואים.
- שימוש נכון ב-MAKEFILE, (במיוחד כאשר אתם נדרשים לחלק את התוכנית למספר קבצים, במסגרת הממ"ן).

- הסתרת אינפורמציה - ושימוש בהפשטת מידע.
- הימנעות ככל שניתן משימוש במשתנים גלובליים.
- שימוש מירבי ונכון במלוא הכלים שמעמידה השפה לרשותכם.
- קוד אלגנטי ולא מסורבל.

ד. יעילות התכנית והתרשמות כללית - 20%

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

ינתנו קנסות במיקרים הבאים:

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

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

לעיתים נוהגים להגיש תוכנית שעושה למעלה מהנדרש. במקרה וכוונתך להגיש תוכנית מעין זו, יש לבדוק כי:

כל הדרישות מהתוכנית המקורית יתקיימו.

כל תוספת תהיה מתועדת היטב.

תוספות המכילות שגיאות עלולות להוריד מהניקוד הסופי, גם אם התוספות לא נדרשו במטלה.

עד כאן ההנחיות, זה הזמן לאחל לך הנאה והצלחה בלימודך.

12. נוהל הגשת מטלות מנחה (ממ"ן)

קיימות שתי חלופות להגשת מטלות:

- **שליחת מטלות באמצעות מערכת המטלות המקוונת**
מערכת שליחת המטלות קלה להפעלה, היא חוסכת את הצורך במילוי טפסים, במשלוח דואר ובשמירת עותק של המטלה, ומאפשרת מעקב אחר המטלה.
הגישה למערכת המטלות המקוונת היא דרך אתר הבית של הקורס בקישור "מערכת המטלות".
- **שליחת מטלות באמצעות הדואר או הגשה ישירה למנחה במפגשי ההנחיה**
לכל מטלת מנחה עליכם לצרף טופס נלווה אחד.
הקפידו למלא את כל הפרטים בחלק א של הטופס. הכניסו את הטופס (על כל חלקיו הצבעוניים) יחד עם המטלה למעטפה המיועדת לכך ורשמו בכתב יד ברור את כתובתכם (כולל מיקוד!) במקום המיועד לכך.
רשמו את שם המנחה וכתובתו באופן מדויק. (דוגמה לטופס נלווה לממ"ן ראו בהמשך).
השאירו עותק של המטלה בידכם!

מועדי הגשה ומשלוח מטלות בדואר

בעמוד הראשון של כל מטלה מצוין מועד הגשתה. יש לשלוח את המטלה עד ל"מועד האחרון להגשה" המצוין עבורה. אסור שחזרת המטלה בדואר על המעטפה תישא תאריך מאוחר מ"המועד האחרון" להגשת הממ"ן.

שימו לב: אין לשלוח מטלות בדואר רשום!
הקפידו לרשום את כתובת המנחה בצורה מדויקת כולל מיקוד.

את הממ"ן עליכם לשלוח לבדיקה **רק למנחה שלקבוצתו אתם משובצים**. ממ"ן שישלח למנחה אחר ללא אישור מראש של מרכז ההוראה ציונו לא ייחשב.

הממ"ן ייבדק ויוחזר לכם תוך שלושה שבועות מהתאריך האחרון להגשת הממ"ן. אם הממ"ן לא יוחזר אליכם במועד זה, אנא התקשרו עם המנחה לבירור סיבת העיכוב.

דחייה בהגשת מטלות

במקרים מיוחדים, כגון שירות מילואים, תוכלו לפנות למנחה שלכם לקבלת אישור לדחיית מועד ההגשה. לכל מטלה המוגשת באיחור צרפו מכתב/אישור המנמק את סיבת האיחור.
בסמכותו של המנחה שלכם לאשר לכם איחור של עד שבוע בהגשת ממ"ן (אלא אם קיבל הנחיות אחרות ממרכז ההוראה). במקרה חריג ביותר שנדרש איחור בהגשה של למעלה מזה יש לבקש אישור של מרכז ההוראה בקורס. מטלות שתגענה באיחור וללא אישור תיבדקנה על-ידי המנחה אך לא יינתן להן ציון והן לא תובאנה בחשבון המטלות המוגשות.

ערעור על ציון בממ"ן

אם יש לכם השגות על הציון שקיבלתם בממ"ן תוכלו להגיש ערעור מנומק בכתב למנחה שלכם בצירוף הממ"ן והטופס המלווה (ההעתק הצהוב), תוך שבוע ימים מיום קבלת הממ"ן.
אם המנחה לא יקבל את ערעורכם, הרשות בידכם לערער בפני מרכז ההוראה בקורס בצירוף הממ"ן והטופס המלווה, תוך שבוע מיום קבלת תשובת המנחה על ערעורכם. החלטת מרכז ההוראה היא סופית.

שימו לב!

את התשובות לממ"נים הנכם מתבקשים לכתוב על דפי פוליו (שורות). כתבו על צדו האחד של העמוד והשאירו שוליים רחבים להערות המנחה (לפחות 5 ס"מ).

האוניברסיטה הפתוחה הקריה ע"ש דורותי דה רוטשילד רח' רבוצקי 108 ת.ד. 808 רעננה 43104		
טופס מלווה למטלה לבדיקה מנחה (ממ"ן)		
לשימוש פנימי		
21	611	1-2 3-7 8-10
חלק א - ימולא על-ידי התלמיד מלא נא את כל הפרטים בעט כדורי בכל המלבנים הכהים וכן למטה. מספר הקורס והמטלה העתק מתוך השאלון. כן הקפד לרשום את כל תשע הספרות של מספר הזהות (גם אפסים וסיפרת ביקורת) שלח את כל העתקים בצירוף המטלה אל מנחה קבוצתך.		
מספר הזהות 1 2 3 4 5 6 7 8 9 11-19	קורס 10125 22-26	מטלה 11 27-28
חלק ג - ציונים יש לרשום מספרים שלמים סכום ציוני השאלות צריך להיות שווה ציון המטלה.		
31 34 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83	ציון שאלה 1 ציון שאלה 2 ציון שאלה 3 ציון שאלה 4 ציון שאלה 5 ציון שאלה 6 ציון שאלה 7 ציון שאלה 8 ציון שאלה 9 ציון שאלה 10 ציון שאלה 11 ציון שאלה 12 ציון שאלה 13 ציון שאלה 14 ציון שאלה 15 ציון שאלה 16 ציון שאלה 17 ציון שאלה 18 ציון שאלה 19 ציון שאלה 20 ציון שאלה 21 ציון שאלה 22 ציון שאלה 23 ציון שאלה 24 ציון שאלה 25	שם התלמיד ישראל ישראלי כתובת התלמיד רח' מנחם 19 ת"א טלפון 03-5269710 שם המנחה ד"ר ארז קבי לימוד 01 מרכז לימוד ת"א 610 נשלח ביום 1.1.02
חלק ב - ימולא על-ידי המנחה מלא נא את כל הפרטים (בעט כדורי). שמור את העותק האחרון בידך. שלח את שאר העותקים בצירוף המטלה למרכז שירות לאוניברסיטה (מש"ל).		
התקבל ביום נשלח ביום שם המנחה		
חלק ד - הערות המנחה לתלמיד (נא כתוב ברור)		

מק"ט 9-830-1-9500 ירחף ורחף ושתי בע"מ

דוגמה למילוי טופס מלווה לממ"ן

שים לב,

עליך להשאיר לעצמך העתק של המטלה.

**אין האוניברסיטה הפתוחה אחראית
למטלה שתאבד בשל תקלות בדואר.**

מטלת מנחה (ממ"ן) 11

הקורס: 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטלה: פרקים 1,2,3

משקל המטלה: 4 נקודות (חובה)

מספר השאלות: 2

מועד אחרון להגשה: 15.11.2009

סמסטר: 2010 א'

קיימות שתי חלופות להגשת מטלות:

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס
 - שליחת מטלות באמצעות הדואר או הגשה ישירה למנחה במפגשי ההנחיה
- הסבר מפורט ב"נוהל הגשת מטלות מנחה"

יש להגיש את קבצי המקור (.c), קבצי ההרצה (.o. וכו'), קבצי הסביבה המתאימים (כולל קבצי MAKEFILE), קבצי קלט וקבצי פלט.

שאלה 1 (תכנית ראשית בקובץ doll. c (45 נקודות)

עליכם לכתב תוכנית הקולטת מהקלט הסטנדרטי, רשימה של מספרים ממשיים, שהראשון בהם הוא השער היציג של הדולר ביום הרצת התוכנית. הערכים הבאים אחריו הם ייצוגים של סכומי כסף בדולרים. על התוכנית לעבור על הקלט, עד לסיומו ולהדפיס בפורמט של טבלה: עבור כל ערך דולרי את הערך השקלי השקול. בשורה התחתונה של הטבלה יש להדפיס את סכום סך הכל הדולרים וסכום סך הכל השקלים שהדפסנו.

לדוגמא: עבור הקלט:

4.10 3.20 11.50 12 3.98

על התוכנית להדפיס את הטבלה הבאה:

<u>\$</u>	<u>IS</u>
12	47.76
11.50	45.77
3.20	12.74

4.10

16.32

30.80

122.58

הערה: לצורך קריאת הנתונים מהקלט ניתן להשתמש בפונקציה scanf .

שאלה 2 (תכנית ראשית בקובץ abc.c) (55 נקודות)

נגדיר קיצורי רצפים באופן הבא: בהינתן שרשרת תווים, נחפש במילה רצפים מתוך האלף-בית האנגלי. לדוגמא בשרשרת הבאה ישנם 3 רצפים: abcemnopqrtaduvwxaz, מודגשים. כל רצף יקוצר לאות הראשונה ברצף, מקף, והאות האחרונה ברצף, לכן עבור הדוגמא למעלה נקבל: da-cemoqm-rrtadu-xaz.

עליכם לכתוב פונקציה המקבלת כפרמטר מערך תווי, ומדפיסה אותו עם קיצורי הרצפים שלו.

בהצלחה!

מטלת מנחה (ממ"ן) 12

הקורס: 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטלה: פרקים (4,5,6)

מספר השאלות: 2 משקל המטלה: 10 נקודות (רשות)

סמסטר: 2010 א' מועד אחרון להגשה: 20.12.2009

קיימות שתי חלופות להגשת מטלות:

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס
 - שליחת מטלות באמצעות הדואר או הגשה ישירה למנחה במפגשי ההנחה
- הסבר מפורט ב"נוהל הגשת מטלות מנחה"

יש להגיש את קבצי המקור (.c), קבצי ההרצה, קבצי הסביבה המתאימים (כולל קבצי MAKEFILE), קבצי קלט וקבצי פלט.

שאלה 1 35 נקודות (תכנית ראשית בקובץ summary.c)

עליכם לכתוב פונקציה המקבלת כפרמטר מערך של מספרים שלמים ואת גודל מערך זה, ומחזירה "סיכום" של המערך, כאשר כל מספר במערך המקורי מוחלף בסכום האיברים עד אליו.

הערה: אסור לפונקציה לגרום לשינויים במערך המקורי.

לדוגמא:

בהינתן המערך הבא:

14 10 5 3

הפונקציה תחזיר את המערך:

33 19 9 8 3

עליכם לכתוב תכנית אינטראקטיבית, הקוראת את המערך ואת גודלו מהקלט הסטנדרטי,

ומדפיסה לפלט הסטנדרטי את ה"סיכום" שלו.

הערה: בפלט ובקלט יש להשתמש בייצוג עשרוני.

שאלה 2 65 נקודות (בקבצים complex.h, complex.c, main.c)

תזכורת מספרים מרוכבים:

מספר מרוכב הוא מספר המורכב ממספר ממשי ומספר מדומה, כאשר ביניהם רשום סימן "+" או "-".

לדוגמא: $a + bi$, כאשר a המספר הממשי ו- bi המספר המדומה.

מספר מדומה מורכב ממכפלה של שני איברים: b מספר ממשי כלשהו ו- i , כאשר i הנו שורש ריבועי של המספר -1.

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

להלן הפעולות החשבוניות הבסיסיות על מספרים מרוכבים:

חיבור בין שני מספרים מרוכבים:

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

חיסור בין שני מספרים מרוכבים:

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

כפל של מספר מרוכב עם מספר ממשי:

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

כפל של מספר מרוכב ומספר מדומה:

$$mi * (a + bi) = mia + mibi = ami + bmii = -bm + ami$$

כפל של מספר מרוכב במספר מרוכב:

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

הערך המוחלט של מספר מרוכב $a + bi$ הוא המספר הממשי החיובי:

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

עליכם לכתוב תכנית מחשב אינטראקטיבית הקוראת פקודות, מפענחת ומבצעת אותן. הפקודות עוסקות בפעולות על מספרים מרוכבים (תזכורת למעלה). עליכם להגדיר, תוך השימוש בפקודת typedef את הטיפוס complex אשר מסוגל להחזיק מספר מרוכב. על מבנה הנתונים שבחרתם להיות יעיל מבחינת כמות זיכרון הנדרשת לשמירתו ויעיל מבחינת הגישה אליו.

בנוסף עליכם להגדיר 6 משתנים חיצוניים A,B,C,D,E,F מטיפוס זה.

כל שם של מספר מרוכב בפקודות שלהלן יילקח מתוך השיטה הנ"ל.

הפקודות המותרות כקלט לתכנית:

1. מספר ממשי, מספר ממשי, שם-מספר-מרוכב read_comp

הפקודה תגרום לקריאת הערכים של המספר המרוכב, לתוך המספר המרוכב ששמו ניתן בפקודה כפרמטר ראשון. מותר לכם להניח שהמספר הממשי הראשון הוא החלק הממשי של המספר המרוכב והמספר הממשי השני הוא המקדם של החלק המדומה של המספר המרוכב.

לדוגמא, הפקודה הבאה :

read_comp A, 5.1, 6.2

מייצגת את המספר המרוכב :

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

2. שם-מספר-מרוכב print_comp

המספר המרוכב ששמו ניתן יודפס בצורה נאה בפלט.

3. חיבור מספרים מרוכבים :

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

תבצע הפעולה הבאה :

מספר-מרוכב-ב' + מספר-מרוכב-א'

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

4. חיסור מספרים מרוכבים :

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

תבצע הפעולה הבאה :

מספר-מרוכב-ב' - מספר-מרוכב-א'

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

5. כפל מספר מרוכב עם מספר ממשי :

מספר ממשי, שם-מספר-מרוכב-א' mult_comp_real

תבצע הפעולה הבאה :

מספר-ממשי * מספר-מרוכב-א'

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

6. כפל מספר מרוכב עם מספר מדומה :

מספר ממשי, שם-מספר-מרוכב-א' mult_comp_img

תתבצע הפעולה הבאה :

$i * \text{מספר-ממשי} * \text{מספר-מרוכב-א'}$

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

7. כפל מספר מרוכב אחד עם מספר מרוכב שני :

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

תתבצע הפעולה הבאה :

$\text{מספר-מרוכב-ב' } * \text{ מספר-מרוכב-א'}$

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

8. חישוב ערך מוחלט של מספר מרוכב :

שם-מספר-מרוכב-א' abs_comp

תתבצע הפעולה הבאה :

$|\text{מספר-מרוכב א'}|$

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

9. **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 בקלט.

הערה: ניתן להיעזר בתוכנית בקובץ תוכניות לדוגמא. לתשומת לבכם, פתרון שיבנה בדומה לפתרון המוצע בקובץ תוכניות לדוגמא, יכלול נושאים המופיעים בפרקים שהם מחוץ לתחום של ממ"ן זה, לכן ניתן כמובן להגיש מימוש פשוט יותר.

דוגמאות:

לפקודה:

read_comp W, 3.2, 8

יש להגיב בהודעה:

"No such complex number"

לפקודה:

kkkk A,B

יש להגיב בהודעה:

"No such command"

לפקודה:

read_comp A, B,567

יש להגיב:

“Wrong parameters, second parameter must be a real number”

וכו'...

דוגמאות לקלט תקין :

```
read_comp A,45.1,23.7
read_comp B,54.2,3.56
print_comp A
add_comp A, B
sub_comp C, A
mul_comp_real A, 2.5
mult_comp_img A, 2.5
mult_comp_comp A, B
abs A
abs B
abs C
halt
```

יש לחלק את התוכנית למספר קבצים : main.c ,complex.c ,ו-complex.h.

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

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

יש לכתוב לתוכנית ממשק סביר, כך שהמשתמש יוכל להבין מה עליו 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) 13

הקורס: 20465 - מעבדה בתכנות מערכות

חומר הלימוד למטלה: פרקים 6,7,8

מספר השאלות: 3 משקל המטלה: 15 נקודות (חובה)

סמסטר: 2010 א' מועד אחרון להגשה: 17.1.2010

קיימות שתי חלופות להגשת מטלות:

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס
 - שליחת מטלות באמצעות הדואר או הגשה ישירה למנחה במפגשי ההנחיה
- הסבר מפורט ב"נוהל הגשת מטלות מנחה"

שאלה 1 (10 נקודות)

בהנחה ש- A ו- B הם מספרים שלמים, עליכם לרשום מה מבצע קטע התוכנית הבא:

$A \wedge B;$

$B \wedge A;$

$A \wedge B;$

יש להסביר בדייקנות. עליכם להשתמש גם בדוגמה, לשם הבהרה.

שאלה 2 (45 נקודות) (תכנית ראשית בקובץ source.c)

תזכורת: כאשר מהדרים (compile) תוכנית בשפת C, הקובץ שנוצר על ידי המהדר (compiler) הוא קובץ בעל שם זהה לקובץ המקור (source), אך בעל סיומת שונה: .exe. כלומר אם כתבנו תוכנית ושמרנו אותה בקובץ בשם name.c, לאחר הידורה נוצר קובץ בשם name.exe. (כמובן שתזכורת זו נכונה עבור תוכנית המוכללת בקובץ יחיד, ולא נכונה עבור תוכנית המחולקת למספר קבצים).

עליכם לכתוב תוכנית המדפיסה את קובץ המקור של עצמה. כלומר אם שם התוכנית המורצת הוא name.exe, על התוכנית להדפיס לפלט הסטנדרטי את תוכן הקובץ name.c.

הערות:

1. הניחו ששני הקבצים, קובץ `c`. וקובץ `exe`, שמורים באותו מדריך (`directory`).
2. ייתכן שבעתיד תאלצו לארגן מחדש את זכרון המחשב שלכם, ותוך כדי הארגון תמצאו לנכון להחליף שמות של קבצים. בהנחה, שהחלפת שמות הקבצים נעשית בצורה מושכלת, כלומר שמות שני הקבצים `exe`. וקובץ `c`. מוחלפים בהתאמה, על התוכנית שלכם להמשיך לעבוד בצורה נכונה, ללא ביצוע שינויים כלשהם. (הכוונה בהחלפה מושכלת, היא שאם שמות הקבצים באופן מקורי היו `name.c` ו-`name.exe` שניהם ישונו באופן זהה. לדוגמא, הם יוחלפו ל-`my_file.c` ול-`my_file.exe`).

שאלה 3 (45 נקודות) (תכנית ראשית בקובץ `min_call.c`)

עליכם לכתוב פונקציה שמקבלת 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 (חובה)

סמסטר : 2010 א' מועד אחרון להגשה : 28.3.2010

קיימות שתי חלופות להגשת מטלות:

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס
- שליחת מטלות באמצעות הדואר או הגשה ישירה למנחה במפגשי ההנחיה

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.

עליך לכתוב תוכנת אסמבלר, לשפת אסמבלי שתוגדר בהמשך. הפרוייקט ייכתב בשפת C. עליך להגיש :

1. קבצי המקור של התוכנית שכתבת (קבצים בעלי סיומת c או h).
2. קבצי הרצה.
3. הגדרת סביבת העבודה (MAKEFILE).
4. דוגמא לקבצי קלט וקבצי הפלט, שנוצרו על ידי הפעלת התוכנית שלך על קבצי קלט אלה.

בגלל גודל הפרוייקט עליך לחלק את התוכנית למספר קבצי מקור. יש להקפיד שהקוד הנמצא בתוכניות המקור יעמוד בקריטריונים של בהירות, קריאות וכתיבה 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 הרלוונטים בנפרד תוך שימוש בפקודת `#define`, ולהימנע ממספרי 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 לכ- 40% ממשקל הפרוייקט.

הפרוייקט כולל כתיבה של תוכנית אסמבלר עבור שפת אסמבלי, שהוגדרה במיוחד עבור פרוייקט זה. מותר לעבוד בזוגות. אין לעבוד בצוות גדול יותר משניים. פרוייקטים שיוגשו בשלישיות או יותר יקבלו ציון אפס. **חובה** ששני סטודנטים, הבוחרים להגיש יחד את הפרוייקט, יהיו שייכים לאותה קבוצה.

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

רקע כללי

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 – 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 התוכנית, לבין שאר הזיכרון.

יחידת העיבוד המרכזית (היע"מ) יכולה לבצע מספר מסוים של הוראות פשוטות, ולשם כך היא משתמשת בכמה אוגרים (register) הקיימים בה. 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 ההוראות 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.

נסביר כיצד מתבצע קוד זה: כל הוראה בקוד יכולה להתייחס לנתון הנמצא בהוראה עצמה, לאוגר או למען בזיכרון. היע"מ מפרקת כל שורת קוד להוראה ולאופרנדים שלה, ומבצעת את ההוראה. אוגר מיוחד בתוך היע"מ מצביע תמיד על ההוראה הבאה לביצוע (program counter). כאשר מגיעה היע"מ להוראת עצירה, היא מחזירה את הפיקוד לתוכנית שהפעילה אותה או למערכת ההפעלה.

לכל שפת תכנות יש, כידוע, מהדר (compiler), או מפרש (interpreter), המתרגם תוכניות מקור לשפת מכונה. אם תוכנית מקור נכתבה בשפת אסמבלי, נקראת התוכנית המתרגמת בשם אסמבלר. בפרוייקט זה עליך לכתוב אסמבלר. לשם כך נעקוב אחרי גלגולה של תוכנית שנכתבה בשפת אסמבלי, שנגדיר במיוחד עבור פרוייקט זה, עד לשליחתה לתוכנת הקישור והטעינה (linker/loader).

לתשומת לבך: בהסברים הכלליים על אופן עבודת תוכנת האסמבלר, יש התייחסות גם לעבודת תוכנת הקישור (linker) ותוכנת הטעינה (loader). התייחסויות אילו הובאו על מנת לאפשר לכם להבין את המשך תהליך העיבוד של הפלט של תוכנת האסמבלר. אל לך לטעות, עליך לכתוב את תוכנת האסמבלר בלבד, אין צורך לכתוב גם את תוכנת הקישור והטעינה!!!

תחילה נגדיר את שפת האסמבלי ואת המחשב הדמיוני שהגדרנו עבור פרוייקט זה.

"חומרה":

המחשב מורכב מיע"מ (יחידת עיבוד מרכזית), אוגרים וזיכרון RAM, כאשר חלק מהזיכרון משמש גם כמחסנית (stack). גודלה של מלת זיכרון במחשב הוא 16 סיביות. האריתמטיקה נעשית בשיטת המשלים ל-2 (2's complement). מחשב זה מטפל רק במספרים שלמים חיוביים ושליליים, אין טיפול במספרים ממשיים.

אוגרים:

למחשב 8 אוגרים כלליים (r0, r1, r2, r3, r4, r5, r6, r7), מונה תוכנית (PC – program counter), מצביע המחסנית של זמן ריצה (SP – stack pointer), ואוגר סטטוס (PSW – program status word) בעל שני דגלים: דגל נשא (Carry) ודגל אפס (Zero). גודלו של כל אוגר הוא 16 סיביות.

עבור ה-PSW הסיביות הראשונות הן C ו-Z, כלומר בתחביר של שפת C :

$$C = (PSW \& 01)$$

$$Z = (PSW \& 02)$$

גודל הזיכרון הוא 2000 מלים (גודל כל מלה 16 סיביות).

קידוד של תווים (characters) נעשה בקוד ascii.

אפיון מבנה הוראת מכונה:

כל הוראת מכונה מורכבת ממילה אחת (בת 16 סיביות), שתי מלים (בנות 16 סיביות כל אחת), או שלוש מלים (בנות 16 סיביות כל אחת). בכל סוגי ההוראות המבנה של המילה הראשונה זהה. מבנה המילה הראשונה בהוראה הוא כדלהלן:

15	12	11	9	8	6	5	3	2	0
קוד ההוראה		שיטת מיעון אופרנד מקור		אוגר אופרנד מקור		שיטת מיעון אופרנד יעד		אוגר אופרנד יעד	

סיביות 12-15 מהוות את קוד ההוראה (opcode). בשפה שלנו יש 16 קודי הוראה והם :

הקוד בבסיס אוקטלי (8)	הקוד בבסיס דצימלי (10)	פעולה
0	0	mov
1	1	cmp
2	2	add
3	3	sub
4	4	mul
5	5	div
6	6	lea
7	7	inc
8	10	dec
9	11	jmp
10	12	bne
11	13	red
12	14	prn
13	15	jsr
14	16	rts
15	17	hlt

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

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

סיביות 3-5 מקודדות את שיטת המיעון של אופרנד היעד (destination operand).

בשפה שלנו קיימות ארבע שיטות מיעון.

חלק משיטות המיעון דורשות מילת מידע נוספת. אם שיטת המיעון של רק אחד משני האופרנדים דורשת מילה נוספת, אזי המילה הנוספת מתייחסת לאופרנד זה. אך אם שיטות המיעון של שני האופרנדים דורשות מילה נוספת אזי המילה הנוספת הראשונה מתייחסת לאופרנד המקור (source operand) והמילה הנוספת השנייה מתייחסת לאופרנד היעד (destination operand).

סיביות 6-8

סיביות אלו מסמלות את מספרו של האוגר (0-7) המשתתף כאופרנד מקור (source) בפעולה. שימו לב! לא כל שיטת מיעון דורשת זהות של אוגר. עבור אותן שיטות מיעון שלא דורשות אוגר, שדה זה לא מנוצל.

סיביות 0-2

סיביות אלו מסמלות את מספרו של האוגר (0-7) המשתתף כאופרנד יעד (destination) בפעולה. שימו לב! לא כל שיטת מיעון דורשת זהות של אוגר. עבור אותן שיטות מיעון שלא דורשות אוגר, שדה זה לא מנוצל.

ארבע שיטות המיעון הקיימות במכונה שלנו הן :

ערך	שיטת מיעון	תוכן המילה נוספת	אוגר	אופן הכתיבה	דוגמא
0	מיעון מידי	המילה הנוספת מכילה את האופרנד עצמו.	סיביות זיהוי האוגר אינן בשימוש בשיטת מיעון זו.	האופרנד מתחיל בתו # ולאחריו ובצמוד אליו מופיע מספר חוקי.	mov #-1,r2
1	מיעון ישיר	המילה הנוספת מכילה מען בזיכרון. תוכן מען זה הינו האופרנד המבוקש.	סיביות זיהוי האוגר אינן בשימוש בשיטת מיעון זו.	האופרנד הינו תווית 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 הנחית 'extern'. (אם התווית הוצהרה כ- external אזי תוכנית הקישור תדאג למתן הערך המתאים).	mov x, r2
3	מיעון עקיף	המילה הנוספת מכילה מען בזיכרון. תוכן מען זה הינו מען בפני עצמו. מען זה (השני) הינו המען של האופרנד המבוקש.	סיביות זיהוי האוגר אינן בשימוש בשיטת מיעון זו.	מיעון עקיף מאופיין על ידי הסימן '@' ובצמוד לאחריו שם תווית. תווית זו מוצהרת או מוגדרת כמו בשיטת מיעון ישיר.	mov @x, r2
4	מיעון אוגר ישיר	אין מילה נוספת בשיטת מיעון זו.	האוגר שמספרו מופיע בשדה זה, מכיל את האופרנד המבוקש.	האופרנד הינו שם חוקי של אוגר.	mov r1,r2

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

מילה שניה ושלישית:

מופיעות באם נדרש בהתאם לשיטות המיעון המוגדרות במילה הראשונה של הפקודה.

אפיון הוראות המכונה:

הוראות המכונה מתחלקות לשלוש קבוצות לפי מספר האופרנדים הדרוש להן.

קבוצה ראשונה:

ההוראות הזקוקות לשני אופרנדים. הפקודות השייכות לקבוצה זו הן:

mov, cmp, add, sub, mul, div, lea

קוד אוקטלי	פקודה	הסבר פעולה	דוגמא	הסבר דוגמא
0	mov	מבצעת העתקה של האופרנד הראשון, אופרנד המקור (source) אל האופרנד השני, אופרנד היעד (destination) (בהתאם לשיטת המיעון).	mov A, r1	העתק תוכן משתנה A לאוגר r1.
1	cmp	מבצעת "השוואה" בין שני האופרנדים שלה. אופן ההשוואה: תוכן אופרנד היעד (השני) מופחת מתוכן אופרנד המקור (הראשון), ללא שמירת תוצאת החיסור. פעולת החיסור מעדכנת את דגל האפס, דגל Z, באוגר הסטטוס, PSW, יודלק, אחרת הוא יאופס.	cmp A, r1	אם תוכן הערך הנמצא במען A זהה לתוכנו של אוגר r1 אזי דגל האפס, Z, באוגר הסטטוס, PSW, יודלק, אחרת הוא יאופס.
2	add	אופרנד היעד (השני) מקבל את סכום אופרנד המקור (הראשון) והיעד (השני).	add A, r0	אוגר r0 מקבל את סכום תוכן משתנה A וערכו הנוכחי של אוגר r0.
3	sub	אופרנד היעד (השני) מקבל את ההפרש בין אופרנד היעד (השני) ואופרנד המקור (הראשון).	sub #3, r1	אוגר r1 מקבל את ההפרש בין תוכן האוגר, r1, והמספר 3.
4	mul	תוכן אופרנד היעד (השני) מקבל את מכפלת אופרנד המקור והיעד.	mul A, r2	אוגר r2 מקבל את תוצאת המכפלה של תוכן משתנה A וערכו הנוכחי של אוגר r2.
5	div	תוכן אופרנד היעד (השני) מקבל את תוצאת חילוק היעד באופרנד המקור.	div A, r2	אוגר r2 מקבל את תוצאת החילוק של ערכו הנוכחי של אוגר r2 בתוכן משתנה A. כלומר: $r2/A$
6	lea	lea – 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 HELLO, r1	המען של תווית HELLO מוכנס לאוגר r1.

קבוצת הפקודות השניה:

הוראות הדורשות אופרנד אחד בלבד. במקרה זה ששת הסיביות (11-6) חסרות משמעות מכיוון שאין אופרנד מקור (אופרנד ראשון) אלא רק אופרנד יעד (שני). על קבוצה זו נמנות ההוראות הבאות:

inc, dec, jmp, bne, red, prn, jsr

קוד אוקטלי	פקודה	הסבר פעולה	דוגמא	הסבר דוגמא
7	inc	הגדלת תוכן האופרנד באחד.	inc r2	$r2 \leftarrow r2 + 1$
10	dec	הקטנת תוכן האופרנד באחד.	dec C	$C \leftarrow C - 1$
11	jmp	קפיצה בלתי מותנית אל המען המיוצג על ידי האופרנד.	jmp LINE	$PC \leftarrow LINE$
12	bne	bne הינו ראשי תיבות של: branch if not equal (to zero). זוהי פקודת הסתעפות מותנית. הערך במצביע התוכנית (PC) יקבל את ערך אופרנד היעד אם ערכו של דגל האפס (דגל Z) באוגר הסטטוס (PSW) הינו 0.	bne LINE	אם ערך דגל Z באוגר הסטטוס (PSW) הינו 0 אזי: $PC \leftarrow LINE$
13	red	קריאה של תו מתוך לוח המקשים אל האופרנד.	red r1	קוד ה-ascii של התו הנקרא מלוח המקשים יוכנס לאוגר r1.
14	prn	הדפסת התו שערך ה-ascii שלו נמצא באופרנד, אל קובץ הפלט הסטנדרטי (stdout).	prn r1	התו אשר קוד ה-ascii שלו נמצא באוגר r1 יודפס לקובץ הקלט הסטנדרטי.
15	jsr	קריאה לשגרה (סברוטניה). הכנסת מצביע התוכנית (PC) לתוך המחסנית של זמן ריצה והעברת ערך האופרנד למצביע התוכנית (PC).	jsr FUNC	$stack[SP] \leq PC$ $SP \leq SP - 1$ $PC \leq FUNC$

קבוצת הפקודות השלישית:

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

ההוראות השייכות לקבוצה זו הן: hlt, rts.

קוד אוקטלי	פקודה	הסבר פעולה	דוגמא	הסבר דוגמא
16	rts	הוראת חזרה משיגרה. ביצוע הוראת pop על המחסנית של זמן ריצה, והעברת הערך שהיה שם לאוגר התוכנית (PC). הוראה זו מורכבת ממלה אחת בלבד (בת 16 סיביות). במלה זו החלק המשמעותי הן הסיביות 15-12 המהוות את קוד ההוראה. לשאר הסיביות אין כל חשיבות.	rts	$SP \leq SP + 1$ $PC \leq stack[SP]$
17	hlt	הוראה לעצירת התוכנית. ההוראה מורכבת ממלה אחת בלבד (בת 16 סיביות). במלה זו החלק המשמעותי הן הסיביות 15-12 המהוות את קוד ההוראה. לשאר הסיביות אין כל חשיבות.	hlt	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	זהו 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. הינו רשימת מספרים חוקיים המופרדים על ידי התו ' (פסיק). למשל:

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

שים 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.

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 הוראת '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’.

להוראת ‘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.

4. ‘extern’.

להוראת ‘extern’ פרמטר אחד בלבד 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 ממשמעות.

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). כמקודם, 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 :

תווית אופציונלית הוראה לדוגמא :

END: hlt

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 במירכאות כפולות(המירכאות 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    LENGTH, r1
           lea     STR, r2
LOOP:      jmp     END
           prn     r2
           sub     #1, r1
           inc     r2
           bne     @LOOP
END:       hlt
STR:       .string "abcdef"
LENGTH:    .data   6

```

עליו להחליף את שמות הפעולה `mov`, `lea`, `jmp`, `prn`, `sub`, `inc`, `bne`, `hlt` בקוד הבינארי השקול להם במחשב שהגדרנו.

כמו כן, על האסמבלר להחליף את הסמלים `STR`, `LENGTH`, `MAIN`, `LOOP`, `END` במענים של שני האתרים שהוקצו לשני הנתונים, ובמענים של ההוראות המתאימות.

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

Label	Decimal Address	Octal Address	Command	Operands	Binary machine code	Octal machine code
(MAIN:)	0100	0144	mov	LENGTH, r1	0000 001 000 100 001	001041
	0101	0145			0000 000 001 111 000	000170
	0102	0146	lea	STR, r2	0110 001 000 100 010	061042
(LOOP:)	0103	0147			0000 000 001 110 001	000161
	0104	0150	jmp	END	1001 000 000 001 000	110010
	0105	0151			0000 000 001 110 000	000160
	0106	0152	prn	r2	1100 000 000 100 010	140042
	0107	0153	sub	#1, r1	0011 000 000 100 001	030041
	0108	0154			0000 000 000 000 001	000001
	0109	0155	inc	r2	0111 000 000 100 010	070042
	0110	0156	bne	@LOOP	1010 000 000 011 000	120030
	0111	0157			1111 111 111 111 010	177772
	0112	0160	hlt		1111 000 000 000 000	170000
(END:)	0113	0161	.string	"abcdef"	0000 000 001 100 001	000141
(STR:)	0114	0162			0000 000 001 100 010	000142
	0115	0163			0000 000 001 100 011	000143
	0116	0164			0000 000 001 100 100	000144
	0117	0165			0000 000 001 100 101	000145
	0118	0166			0000 000 001 100 110	000146
	0119	0167			0000 000 000 000 000	000000
(LENGTH:)	0120	0170	.data	6	0000 000 000 000 110	000006

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 0104 (עשרוני או 150 אוקטלי), אלא רק לאחר שהתוכנית נקראה.

אי לכך יש שתי פעולות נפרדות שצריך לבצע לגבי כל הסמלים שהוגדרו ביד המתכנת. הראשונה היא לבנות טבלה של כל הסמלים והערכים המספריים המשויכים להם, והשנייה היא להחליף את כל הסמלים, המופיעים בתוכנית בשדה המען, בערכיהם המספריים. שלבים אלה קשורים בשתי סריקות, מעברים, של קוד המקור. במעבר הראשון נבנית טבלת סמלים בזיכרון, שמותאמים בה מענים לכל הסמלים. בדוגמא דלעיל, טבלת הסמלים לאחר מעבר ראשון היא:

ערך אוקטלי	ערך דצימלי	סמל
0144	0100	MAIN
0150	0104	LOOP
0160	0112	END
0161	0113	STR
0170	0120	LENGTH

במעבר השני נעשית ההחלפה כדי לתרגם את הקוד לבינארי. עד אותו זמן צריכים הערכים של כל הסמלים להיות כבר ידועים.

שים לב, שני המעברים של האסמבלר נעשים עוד לפני שתוכנית המשתמש נטענת לזיכרון לצורך הביצוע: כלומר, התרגום נעשה בזמן אסמבלי. שהוא הזמן שבו נמצאת הבקרה בידי האסמבלר.

לאחר השלמת תהליך התרגום יכולה תוכנית המשתמש לעבור לשלב הקישור/טעינה ולאחר מכן לשלב הביצוע. הביצוע נעשה בזמן ריצה.

המעבר הראשון

במעבר הראשון נדרשים כללים כדי לקבוע איזה ערך מען ישויד לכל סמל. העיקרון הבסיסי הוא לספור את המקומות בזיכרון שאותם צורכת כל הוראה כאשר היא נקראת. אם כל הוראה תיטען לאחר האסמבלי לאתר העוקב של אתר ההוראה הקודמת, תציין ספירה כזאת את מען ההוראה. הספירה נעשית על ידי האסמבלר ומוחזקת באוגר הנקרא מונה אתרים. 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 לעיל, 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 איננה כה פשוטה. יש הרבה הוראות המשתמשות בצורות מיעון שונות. אותה הוראה יכולה לקבל משמעויות שונות בכל אחת מצורות המיעון, ולכן יתאימה לה קודים שונים. לדוגמא, 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.

כך גם במקרה שלנו.

סיביות 12-15 במילה הראשונה של ההוראה מייצגות את קוד ההוראה, וסיביות 3-5 וסיביות 9-11 מייצגות שיטות מיעון.

במחשב שלנו קיימת גמישות לגבי שיטת המיעון של שני האופרנדים. הערה: דבר זה לא מחייב לגבי כל מחשב. ישנם מחשבים שכל 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. לדוגמא: 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 עתה כיצד נפתרת בעיה זו.

מעבר שני

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.

אסמבלר של מעבר אחד

יש תוכניות אסמבלר שאינן מבצעות את המעבר השני, והחלפת המענים נעשית בהם בדרך הבאה: בזמן המעבר הראשון שומר האסמבלר טבלה שבה נשמר עבור כל תווית, מען ההוראה שיש בה התייחסות אל התווית בחלק המען.

דוגמא:

נניח שבמען 400 בתוכנית מוגדרת התווית TAB.

נניח גם שבמען 300 מופיע add TAB, r1.

ובמען 500 מופיע jmp TAB.

```
300:      add    TAB, r1
```

```
.....  
400:  TAB:  .....
```

```
.....  
500:      jmp    TAB
```

האסמבלר יקצה כניסה בטבלה לתווית TAB, ויניח בה את המענים 301 ו-501. (המענים הנשמרים הם 301 ו-501 ולא 300 ו-500 מכיוון ששורת ההוראה מופיעה בכתובות אלה, והמילה

הנוספת מופיעה בכתובת שבאה מיד לאחר מכך). (המענים יכולים להישמר גם בכניסות נפרדות, הדבר תלוי בצורת היישום). בסוף המעבר הראשון ימלא האסמבלר את המענים החסרים בתרגום הקוד מתוך הטבלה. היתרון בשיטה זו הוא כמובן, שהאסמבלר חוסך את המעבר השני על כל התוכנית.

הפרדת הוראות ונתונים

לכמה תוכניות אסמבלר יש מוני אתרים אחדים. אחד השימושים לכך הוא הפרדת הקוד והנתונים לקטעים שונים בזיכרון, שיטה שהיא עדיפה על פני הצמדה של הגדרות הנתונים להוראות המשתמשות בהן.

אחת הסכנות הטמונות באי הפרדת הקוד מהנתונים היא, שלפעמים עלול המעבד, בעקבות שגיאה קלה, לנסות לבצע את הנתונים. שגיאה שיכולה לגרום זאת היא, למשל, השמטת הוראת עצירה או הסתעפות לא נכונה. אם הנתון שאותו מנסה המעבד לבצע אינו מהווה קוד של הוראה חוקית, תתקבל מיד הודעת שגיאה. אך אילו הנתון נראה כהוראה חוקית, הייתה הבעיה חמורה יותר, משום שהשגיאה לא הייתה מתגלית מיד.

בתוכנית האסמבלר, שעליך לממש, יש להפריד בין קטע הנתונים לקטע ההוראות.

גילוי שגיאות אסמבלר

האסמבלר יכול לגלות שגיאות בתחביר של השפה, כגון הוראה שאינה קיימת, מספר אופרנדים שגוי, אופרנד שאינו מתאים להוראה ועוד. כן מוודא האסמבלר שכל הסמלים מוגדרים פעם אחת בדיוק. מכאן שאת השגיאות המתגלות בידי האסמבלר אפשר לשייך בדרך כלל לשורת קלט מסוימת. אם, לדוגמה, ניתנו שני מענים בהוראה שאמור להיות בה רק מען יחיד, האסמבלר עשוי לתת הודעת שגיאה בנוסח "יותר מדי מענים". בדרך כלל מודפסת הודעה כזאת בתדפיס הפלט באותה שורה או בשורה הבאה, אם כי יש תוכניות אסמבלר המשתמשות בסימון מקוצר כלשהו, ומפרטות את השגיאות בסוף התוכנית.

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

פעולה	הקוד בבסיס אוקטלי	שיטות מיעון חוקיות עבור אופרנד מקור	שיטות מיעון חוקיות עבור אופרנד יעד
mov	0	0,1,3,4	1,3,4
cmp	1	0,1,3,4	0,1,3,4
add	2	0,1,3,4	1,3,4
sub	3	0,1,3,4	1,3,4
mul	4	0,1,3,4	1,3,4
div	5	0,1,3,4	1, 3, 4
lea	6	1	1,3,4
inc	7	אין אופרנד מקור	1,3,4
dec	10	אין אופרנד מקור	1,3,4
jmp	11	אין אופרנד מקור	1,3
bne	12	אין אופרנד מקור	1,3
red	13	אין אופרנד מקור	1,3,4
prn	14	אין אופרנד מקור	0,1,3,4
jsr	15	אין אופרנד מקור	1,3
rts	16	אין אופרנד מקור	אין אופרנד יעד
hlt	17	אין אופרנד מקור	אין אופרנד יעד

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

אלגוריתם כללי

להלן נציג אלגוריתם כללי למעבר הראשון ולמעבר השני: נניח כי הקוד מחולק לשני אזורים, אזור ההוראות (code) ואזור הנתונים (data). נניח כי 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. $DC \leq 0, IC \leq 0$.
2. קרא שורה.
3. האם השדה הראשון הוא סמל? אם לא, עבור ל-5.
4. הצב דגל "יש סמל".
5. האם זוהי הוראה מדומה (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 data). ערכו יהיה DC.
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 code). ערכו יהיה IC.
12. חפש בטבלת ההוראות, אם לא מצאת – הודע על שגיאה בקוד ההוראה.
13. $IC \leq L + IC$.
14. חזור ל-2.

מעבר שני

1. $IC \leq 0$.
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. $IC \leq IC + L$.
10. חזור ל-2.
11. שמור בקובץ 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  LENGTH, r1
           lea  STR, r2
LOOP:      jmp  END
           prn  r2
           sub  #1, r1
           inc  r2
           bne  @LOOP
END:        hlt
STR:        .string "abcdef"
LENGTH:    .data 6

```

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. הקוד יוטען החל מהמען 0.

Label	Decimal Address	Octal Address	Command	Operands	Binary machine code	Octal machine code
(MAIN::)	0100	0144	mov	LENGTH, r1	0000 001 000 100 001	001041
	0101	0145			LENGTH	????
	0102	0146	lea	STR, r2	0110 001 000 100 010	061042
(LOOP:)	0103	0147			STR	????
	0104	0150	jmp	END	1001 000 000 001 000	110010
	0105	0151			END	????
	0106	0152	prn	r2	1100 000 000 100 010	140042
	0107	0153	sub	#1, r1	0011 000 000 100 001	030041
	0108	0154			0000 000 000 000 001	000001
	0109	0155	inc	r2	0111 000 000 100 010	070042
	0110	0156	bne	@LOOP	1010 000 000 011 000	120030
	0111	0157			1111 111 111 111 010	177772
	0112	0160	hlt		1111 000 000 000 000	170000
(END:)	0113	0161	.string	"abcdef"	0000 000 001 100 001	000141
(STR:)	0114	0162			0000 000 001 100 010	000142
	0115	0163			0000 000 001 100 011	000143
	0116	0164			0000 000 001 100 100	000144
	0117	0165			0000 000 001 100 101	000145
	0118	0166			0000 000 001 100 110	000146
	0119	0167			0000 000 000 000 000	000000
	0120	0170	.data	6	0000 000 000 000 110	000006

טבלת הסמלים :

סמל	ערך דצימלי	ערך אוקטלי
MAIN	0100	0144
LOOP	0104	0150
END	0112	0160
STR	0113	0161
LENGTH	0120	0170

נבצע עתה את המעבר השני ונרשום את הקוד בצורתו הסופית :

Label	Decimal Address	Octal Address	Command	Operands	Binary machine code	Octal machine code
(MAIN::)	0100	0144	mov	LENGTH, r1	0000 001 000 100 001	001041
	0101	0145			0000 000 001 111 000	000170
	0102	0146	lea	STR, r2	0110 001 000 100 010	061042
(LOOP:)	0103	0147			0000 000 001 110 001	000161
	0104	0150	jmp	END	1001 000 000 001 000	110010
	0105	0151			0000 000 001 110 000	000160
	0106	0152	prn	r2	1100 000 000 100 010	140042
	0107	0153	sub	#1, r1	0011 000 000 100 001	030041
	0108	0154			0000 000 000 000 001	000001
	0109	0155	inc	r2	0111 000 000 100 010	070042
	0110	0156	bne	@LOOP	1010 000 000 011 000	120030
	0111	0157			0000 000 001 100 000	000150
	0112	0160	hlt		1111 000 000 000 000	170000
(END:)	0113	0161	.string	"abcdef"	0000 000 001 100 001	000141
(STR:)	0114	0162			0000 000 001 100 010	000142
	0115	0163			0000 000 001 100 011	000143
	0116	0164			0000 000 001 100 100	000144
	0117	0165			0000 000 001 100 101	000145
	0118	0166			0000 000 001 100 110	000146
	0119	0167			0000 000 000 000 000	000000
	0120	0170	.data	6	0000 000 000 000 110	000006

לאחר סיום עבודת תוכנית האסמבלר, התוכנית נשלחת אל תוכנית הקישור/טעינה.

תפקידיה של תוכנית זו הן :

1. להקצות מקום בזיכרון עבור התוכנית (allocation).
2. לגרום לקישור נכון בין הקבצים השונים של התוכנית (linking).
3. לשנות את כל המענים בהתאם למקום הטעינה (relocation).
4. להטעין את הקוד פיסית לזיכרון (loading).

לא נדון כאן בהרחבה באופן עבודת תוכנית הקישור/טעינה.

לאחר עבודת תוכנית זו, התוכנית טעונה בזיכרון ומוכנה לריצה.

כעת נעיר מספר הערות ספציפיות לגבי המימוש שלכם :

על תוכנית האסמבלר שלכם לקבל כארגומנטים של שורת פקודה (command line arguments) רשימה של קבצי טקסט בהם רשומות הוראות בתחביר של שפת האסמבלי שהוגדרה למעלה. עבור כל קובץ יוצר האסמבלר קובץ מטרה (object). כמו כן ייווצר (עבור אותו קובץ) קובץ externals באם המקור (source) הצהיר על משתנים מסיימים כעל נקודות כניסה.

קבצי המקור של האסמבלר 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 תקרא את הקבצים : x.as, y.as, hello.as

האסמבלר יוצר את 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 יתואר בהמשך.

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

העבודה על קובץ מסוים 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 – 16 סיביות). במערך הקוד מכניס האסמבלר את הקידוד של הוראות המכונה בהן הוא נתקל במהלך האסמבלי. במערך הנתונים מכניס האסמבלר נתונים המתקבלים תוך כדי האסמבלי (על ידי 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 טבלת סמלים אשר בה נשמרים המשתנים בהם נתקל האסמבלר במהלך ריצתו על הקובץ. לכל משתנה נשמרים שמו, ערכו וטיפוסו (external או relocatable).

אופן פעולת האסמבלר

האסמבלר קורא את קובץ המקור שורה אחר שורה, מחליט מהו טיפוס 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 הוא מספר האוגר.
- אם זו תווית – ערכו הוא הערך שלה כפי שהוא מופיע בטבלת הסמלים.
- אם זה מספר (מיעון ישיר) – ערכו הוא הערך של המספר.

קביעת שיטת המיעון נעשית בהתאם לתחביר של האופרנד כפי שהוא מתואר בחלק העוסק בשיטות המיעון. ככלל התו # מציין מיעון מיידי, תווית מציינת מיעון ישיר, שם של אוגר מציין מיעון אוגר, @ צמודה לשם של משתנה מציין מיעון עקיף.

שימו לב: ערך שדה האופרנד הינו ערך תווית המשתנה כפי שהוא מופיע בטבלת הסמלים.

לאחר שהאסמבלר החליט לגבי הדברים 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: 0, 1 או 2 ומגדיל את מונה הקוד בהתאם (1, 2 או 3 בהתאמה).

דוגמא:

אם ההוראה הייתה

cmp #-3, r1

אזי במלה הראשונה במערך הקוד יאוחסן הערך (בספרות בינאריות):

0001 000 000 100 001

או בספרות אוקטליות:

010041

במקום השני במערך הקוד יאוחסן הערך (ספרות בינאריות):

1111 111 111 111 101

או בספרות אוקטליות:

177775

שהוא הערך 3- בשיטת המשלים ל-2 עבור מלה בגודל של 16 סיביות.

אם ההוראה היא בעלת אופרנד אחד בלבד, כלומר האופרנד הראשון (אופרנד המקור) אינו מופיע, אזי התרגום הינו זהה לחלוטין (ההוראה אף עשויה לתפוס שתי מלים בזיכרון) למעט העובדה שסיביות 6-11 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, hlt) 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 לסוגה 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 הינה 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 נתנה 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 אפס בזיכרון, קידוד ההוראה השניה למען שלאחר ההוראה הראשונה (מען 1 או 2 או 3, 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 ראשוניים הם הנתונים המופיעים ראשוניים בקובץ האסמבלי, וכך הלאה. התייחסות בקובץ האסמבלי למשתנה שהוגדר בקובץ תקודד כך שתצביע על המקום המתאים בתמונת הזיכרון שבונה האסמבלר. פורמט של קובץ object הוא תמונת הזיכרון הנ"ל.

קובץ object מורכב משורות שורות של טקסט, השורה הראשונה מכילה (בצורה אוקטלית) את אורך הקוד (במילות זיכרון) ואת אורך הנתונים (במילות זיכרון). שני המספרים מופרדים ביניהם על ידי רווח. השורות הבאות מתארות את תוכן הזיכרון (שוב, בצורה אוקטלית) החל במען אפס וכלה במען (1 - גודל הנתונים + גודל הקוד).

בהמשך מופיע קובץ object לדוגמא ששמו: ps.obj המתאים ל-ps.as (המספרים המופיעים שם הם: 24, גודל הקוד, ו-12, גודל הנתונים. המספרים הם מספרים אוקטליים, כלומר בסיס 8).

בדוגמא שבהמשך, הקוד ימצא בין המענים 0 ל-23 והנתונים יימצאו החל ממען 24 ועד למען **12-24**. (שוב, כל המספרים אוקטליים).

בנוסף, עבור כל תא זיכרון המכיל הוראה (לא data) מופיע מידע עבור תכנית הקישור. מידע זה הוא אחד משלושת התווים 'e' 'a' או 'r'.

האות 'a' מציינת את העובדה שתוכן התא הוא אבסולוטי (absolute) ואינו תלוי היכן באמת יטען הקובץ (האסמבלר יוצא מתוך הנחה שהקובץ נטען החל ממען אפס).

האות 'r' מציינת שתוכן תא הזיכרון הינו relocatable ויש להוסיף לתוכן התא את ההיסט (Offset) המתאים (בהתאם למקום בו יטען הקובץ באופן מעשי). ה-offset הינו מען הזיכרון שבו תטען למעשה ההוראה אשר האסמבלר אומר שעליה להיטען במען אפס.

האות 'e' מציינת שתוכן תא הזיכרון תלוי במשתנה חיצוני external וכי תכנית הקישור תדאג להכנסת הערך המתאים.

קובץ ה-entries

קובץ ה-entries בנוי משורות טקסט. כל שורה מכילה את שם ה-entry וערכה, כפי שחושב עבור אותו קובץ (ראה לדוגמא את הקובץ ps.ent המתאים לקובץ האסמבלי ששמו ps.as).

קובץ externals

קובץ ה-externals בנוי אף הוא משורות טקסט. כל שורה מכילה את שם ה-external ואת המען בזיכרון שבו יש התייחסות למשתנה חיצוני זה. למשל בקובץ ps.as שבהמשך מופיעה ההוראה:

```
jsr    REVERSE
```

האופרנד של ההוראה (REVERSE) הוא חיצוני. ערכו יצטרך להיכנס למקום השבע עשרה בתמונת הזיכרון אותה מכין האסמבלי עבור אותו הקובץ. לכן, בין היתר, מופיעה בקובץ ה-externals ששמו ps.ext השורה:

```
REVERSE    20
```

קבצי דוגמא:

ארבעת הקבצים ps.as, cs.as, rs.as, a.as שלהלן מהווים תוכנית אחת.

; file ps.as – includes main routine of reversing string “abcdef”

	.entry	STRADD
	.entry	MAIN
	.extern	REVERSE
	.extern	PRTSTR
	.extern	COUNT
STRADD:	.data	0
STR:	.string	“abcdef”
LASTCHAR:	.data	0
LEN:	.data	0

; count length of string, print the original string, reverse string, print reversed string.

```
MAIN:      lea      STR, STRADD
           jsr      COUNT
           jsr      PRTSTR
           mov      @STRADD, LASTCHAR
           add      LEN, LASTCHAR
           dec      LASTCHAR
           jsr      REVERSE
           jsr      PRTSTR
           hlt
```

קובץ cs.as :

;file cs.as - includes count.

```
                .entry      COUNT
                .extern     STRADD
                .extern     LEN
COUNT:         mov        STRADD, r1
                cmp        #0, r1
                jmp        ENDCOUNT
                inc        LEN
                inc        r1
                bne        COUNT
ENDCOUNT:     rts
```

קובץ a.as :

;file a.as – includes routine PRTSTR. This routine prints a string

```
                .entry      PRTSTR
PRTSTR:         mov        STRADD, r2
```

```

LOOP:      cmp      #0, r2
           jmp      BYE
           prn      r2
           inc      r2
           bne      LOOP
BYE:       rts
           .extern  STRADD

```

קובץ rs.as :

;file rs.as - includes routine reverse

```

           .entry   REVERSE
REVERSE:   mov      STRADD, r1
           mov      LASTCHAR ,r2
           mov      LEN, r3

```

;Exchanging places

```

LOOP:      cmp      #0, r3
           jmp      END
           cmp      r1, r2
           jmp      END
           mov      r1, r4
           mov      r2, r1
           mov      r4, r2
           sub      #2, r3
           bne      LOOP
END:       rts
           .extern  LASTCHAR
           .extern  STRADD

```

הקבצים שלהלן הם קבצי object ששמותיהם בעלי סיומת '.ob'. אלו הם הקבצים שהתקבלו מהפעלת התוכנית assembler על קבצי האסמבלר דלעיל. כל תוכן הקבצים מיוצג על ידי מספרים אוקטליים. נתחיל עם קובץ ps.ob :

Label	Decimal Address	Octal Address	Command	Operands	Binary machine code	Octal machine code	Absolute, relocatable or external
						24 12	
(MAIN:)	0000	0000	lea	STR,STRADD	0110 001 000 001 000	061010	a
	0001	0001			0000 000 000 010 101	000025	r
	0002	0002			0000 000 000 010 100	000024	r
	0003	0003	jsr	COUNT	1101 000 000 001 000	150010	a
	0004	0004			0000 000 000 000 000	000000	e
	0005	0005	jsr	PRTSTR	1101 000 000 001 000	150010	a
	0006	0006			0000 000 000 000 000	000000	e
	0007	0007	mov	@STRADD, LASTCHAR	0000 011 000 001 000	003010	a
	0008	0010			0000 000 000 010 100	000024	a
	0009	0011			0000 000 000 011 100	000034	r
	0010	0012	add	LEN, LASTCHAR	0010 001 000 001 000	021010	a
	0011	0013			0000 000 000 011 101	000035	r
	0012	0014			0000 000 000 011 100	000034	r
	0013	0015	dec	LASTCHAR	1000 000 000 001 000	100010	a
	0014	0016			0000 000 000 011 100	000034	r
	0015	0017	jsr	REVERSE	1101 000 000 001 000	150010	a
	0016	0020			0000 000 000 000 000	000000	e
	0017	0021	jsr	PRTSTR	1101 000 000 001 000	150010	a
	0018	0022			0000 000 000 000 000	000000	e
	0019	0023	hlt		1111 000 000 000 000	170000	a
(STRADD:)	0020	0024	.data	0	0000 000 000 000 000	000000	
(STR:)	0021	0025	.string	"abcdef"	0000 000 001 100 001	000141	
	0022	0026			0000 000 001 100 010	000142	
	0023	0027			0000 000 001 100 011	000143	
	0024	0030			0000 000 001 100 100	000144	
	0025	0031			0000 000 001 100 101	000145	
	0026	0032			0000 000 001 100 110	000146	
	0027	0033			0000 000 000 000 000	000000	
(LASTCHAR:)	0028	0034	.data	0	0000 000 000 000 000	000000	
(LEN:)	0029	0035	.data	0	0000 000 000 000 000	000000	

כלומר תוכן קובץ ps.ob הוא:

Octal Address	Octal machine code	Absolute, relocatable or external
	24 12	
0000	061010	a
0001	000025	r
0002	000024	r
0003	150010	a
0004	000000	e
0005	150010	a
0006	000000	e
0007	003010	a
0010	000015	a
0011	000034	r
0012	021010	a
0013	000035	r
0014	000034	r
0015	100010	a
0016	000034	r
0017	150010	a
0020	000000	e
0021	150010	a
0022	000000	e
0023	170000	a
0024	000000	
0025	000141	
0026	000142	
0027	000143	
0030	000144	
0031	000145	
0032	000146	
0033	000000	
0034	000000	
0035	000000	

קובץ: cs.ob

Label	Decimal Address	Octal Address	Command	Operands	Binary machine code	Octal machine code	Absolute, relocatable or external
						14 0	
(COUNT:)	0000	0000	mov	STRADD,r1	0000 001 000 100 001	001041	a
	0001	0001			0000 000 000 000 000	000000	e
	0002	0002	cmp	#0, r1	0001 000 000 100 001	010041	a
	0003	0003			0000 000 000 000 000	000000	a
	0004	0004	jmp	ENDCOUNT	1001 000 000 001 000	110010	a
	0005	0005			0000 000 000 001 011	000013	r
	0006	0006	inc	LEN	0111 000 000 001 000	070010	a
	0007	0007			0000 000 000 000 000	000000	e
	0008	0010	inc	r1	0111 000 000 100 001	070041	a
	0009	0011	bne	COUNT	1010 000 000 001 000	120010	a
	0010	0012			0000 000 000 000 000	000000	r
(ENDCOUNT:)	0011	0013	rts		1110 000 000 000 000	160000	a

כלומר תוכן קובץ cs.ob יהיה:

Octal Address	Octal machine code	Absolute, relocatable or external
	14 0	
0000	001041	a
0001	000000	e
0002	010041	a
0003	000000	a
0004	110010	a
0005	000013	r
0006	070010	a
0007	000000	e
0010	070041	a
0011	120010	a
0012	000000	r
0013	160000	a

קובץ a.ob:

Label	Decimal Address	Octal Address	Command	Operands	Binary machine code	Octal machine code	Absolute, relocatable or external
						13 0	
(PRTSTR:)	0000	0000	mov	STRADD,r2	0000 001 000 100 010	001042	a
	0001	0001			0000 000 000 000 000	000000	e
(LOOP:)	0002	0002	cmp	#0, r2	0001 000 000 100 010	010042	a
	0003	0003			0000 000 000 000 000	000000	a
	0004	0004	jmp	BYE	1001 000 000 001 000	110010	a
	0005	0005			0000 000 000 001 010	000012	r
	0006	0006	prn	r2	1100 000 000 100 010	140042	a
	0007	0007	inc	r2	0111 000 000 100 010	070042	a
	0008	0010	bne	LOOP	1010 000 000 001 000	120010	a
	0009	0011			0000 000 000 000 010	000002	r
(BYE:)	0010	0012	rts		1110 000 000 000 000	160000	a

כלומר תוכן קובץ a.ob יהיה:

Octal Address	Octal machine code	Absolute, relocatable or external
	13 0	
0000	001042	a
0001	000000	e
0002	010042	a
0003	000000	a
0004	110010	a
0005	000012	r
0006	140042	a
0007	070042	a
0010	120010	a
0011	000002	r
0012	160000	a

rs.ob: קובץ

Label	Decimal Address	Octal Address	Command	Operands	Binary machine code	Octal machine code	Absolute, relocatable or external
						25 0	
(REVERSE:)	0000	0000	mov	STRADD,r1	0000 001 000 100 001	001041	a
	0001	0001			0000 000 000 000 000	000000	e
	0002	0002	mov	LASTCHAR,r2	0000 001 000 100 010	001042	a
	0003	0003			0000 000 000 000 000	000000	e
	0004	0004	mov	LEN,r3	0000 001 000 100 011	001043	a
	0005	0005			0000 000 000 000 000	000000	e
(LOOP:)	0006	0006	cmp	#0,r3	0001 000 000 100 011	010043	a
	0007	0007			0000 000 000 000 000	000000	a
	0008	0010	jmp	END	1001 000 000 001 000	110010	a
	0009	0011			0000 000 000 010 100	000024	r
	0010	0012	cmp	r1,r2	0001 100 001 100 010	014142	a
	0011	0013	jmp	END	1001 000 000 001 000	110010	a
	0012	0014			0000 000 000 010 100	000024	r
	0013	0015	mov	r1,r4	0000 100 001 100 100	004144	a
	0014	0016	mov	r2,r1	0000 100 010 100 001	004241	a
	0015	0017	mov	r4, r2	0000 100 100 100 010	004442	a
	0016	0020	sub	#2,r3	0011 000 000 100 011	030043	a
	0017	0021			0000 000 000 000 010	000002	a
	0018	0022	bne	LOOP	1010 000 000 001 000	120010	a
	0019	0023			0000 000 000 000 110	000006	r
(END:)	0020	0024	rts		1110 000 000 000 000	160000	a

כלומר תוכן קובץ rs.ob יהיה:

Octal Address	Octal machine code	Absolute, relocatable or external
	25 0	
0000	001041	a
0001	000000	e
0002	001042	a
0003	000000	e
0004	001043	a
0005	000000	e
0006	010043	a
0007	000000	a
0010	110010	a
0011	000024	r
0012	014142	a
0013	110010	a
0014	000024	r
0015	004144	a
0016	004241	a
0017	004442	a
0020	030043	a
0021	000002	a
0022	120010	a
0023	000006	r
0024	160000	a

קבצי ה-entries המתאימים הם:

		<u>קובץ ps.ent:</u>
MAIN	0	
STRADD	24	
		<u>קובץ cs.ent:</u>
COUNT	0	
		<u>קובץ a.ent:</u>
PRTSTR	0	
		<u>קובץ rs.ent:</u>
REVERSE	0	

קובץ: ps.ext

COUNT	4
PRTSTR	6
REVERSE	20
PRTSTR	22

קובץ: cs.ext

STRADD	1
LEN	7

קובץ: a.ext

STRADD	1
--------	---

קובץ: rs.ext

STRADD	1
LASTCHAR	3
LEN	5

לתשומת לבך : אם בקובץ מסויים אין הצהרת *extern*. אזי לא ייוצר עבורו קובץ *ext*. המתאים.
כנ"ל עבור קבצים שאינם מכילים הודעות *entry*, במקרה זה לא ייוצר קובץ *ent*. מתאים.

סיכום והנחיות כלליות

- אורך התוכנית, הניתנת כקלט לאסמבלר, אינו ידוע מראש (ואינו קשור לגודל 2000 – 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.
- 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 שיווצרו 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 הפרוייקט.

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 ופתרון הממ"נים, והתשובות יכולות לעזור לכולם.
לתשומת לבכם, לא יתקבלו ממ"נים באיחור ללא תיאום מראש עם מרכזת הקורס. ממ"נים שיוגשו באיחור ללא אישור, יקבלו ציון 0.

בהצלחה!!!!

