|  |  |
| --- | --- |
| **מאפייני מחשב** | מעבד עם רגיסטר PSW המכיל מספר דגלים המאפיינים את מצב הפעילות במעבד בכל רגע נתון. אוגרים (r0-7) פריפיקס: @ זיכרון - 1024 בתים (0-1023) קידוד ההוראה הראשונה מתחיל במען 100 (כלומר 0-99 תפוסים).  12 סיביות (0-11)  מספרים שלמים חיוביים ושליליים בלבד, ללא תמיכה בממשיים.  2's complement  תווים בקוד ascii |
| **מבנה הוראת מכונה** | מילה אחת עד 3 בגג. מילה ראשונה היא הפקודה, השאר מייצגות אופרנדים. מבנה מילה ראשונה זהה עבור כל סוג פקודה. מבנה בינארי:   |  |  |  |  | | --- | --- | --- | --- | | 1 0 | 4 3 2 | 8 7 6 5 | 11 10 9 | | A,R,E | מיעון אופרנד יעד | Opcode | מיעון אופרנד מקור |   קידוד המילה בסוף ייעשה בבסיס base64 opcode זה הקוד המייצג את הפקודות הקיימות, אלו הפקודות:    A,R,E - קידוד מוחלט (Absolute), חיצוני (External ) או מצריך מיקום מחדש (Relocatable). A – 00 - תוכן המילה **לא תלוי** במקום בו בזיכרון בו ייטען בפועל קוד המכונה של התוכנית בעת ביצועה (למשל אופרנד מיידי). E - 01 - תוכן המילה **תלוי** בסמל חיצוני (למשל מילת כתובת של תווית extern).  R - 10 - תוכן המילה **תלוי** במקום בו בזיכרון בו ייטען בפועל קוד המכונה של התוכנית בעת ביצועה (למשל מילת כתובת של תווית מקובץ המקור).  סיביות אלה מתווספות רק לקידודים של הוראות (לא של נתונים), והן מתווספות גם לכל המילים הנוספות שיש לקידודים אלה.  שיטות מיעון:   |  |  |  |  | | --- | --- | --- | --- | | קוד | שיטת מיעון | תוכן מילים נוספות | אופן כתיבה | | 1 | מידי Imm | מילה נוספת, נתון, 10 סיביות + ARE | מספר שלם בעשרוני ללא פריפיקס | | 3 | ישיר Directive | מילה נוספת, כתובת, 10 סיביות + ARE | תווית (שהוצהרה או תוצהר) | | 5 | אוגר ישיר Register | מילה נוספת, סיביות 2-6 אוגר יעד, 7-11 אוגר מקור + ARE | שם של אוגר, עם פריפיקס @ | |
| **מפרט הוראות מכונה** | PC - רגיסטר מעבד (לא רגיסטר כללי) שמכיל בכל רגע נתון את הכתובת של מילת הזיכרון הנוכחית שמתבצעת (המילה הראשונה).  קבוצות הוראות המכונה:  1. פקודות בעלות 2 אופרנדים:    2. פקודות בעלות אופרנד בודד:      סיביות 9-11 יכילו תמיד 0 כי אין אופרנד מקור אלא רק יעד.  3. פקודות ללא אופרנד:  בעלות מילה אחת בלבד (המילה הראשונה).  מיעון האופרנדים: 0. |
| **מבנה תכנית בשפת אסמבלי** | מאקרו: מתחיל במילה mcro ובשם המאקרו  מסתיים במילה endmcro  שימוש במאקרו - אזכור שם המאקרו  התוכנית לאחר פרישת המקרו היא התוכנית שהאסמבלר אמור לתרגם.  • אין במערכת הגדרות מאקרו מקוננות.  • שם של הוראה או הנחיה לא יכול להיות שם של מאקרו.  • ניתן להניח שלכל שורת מאקרו בקוד המקור קיימת סגירה עם שורת endmcro (אין צורך לבדוק זאת)  • הגדרת מאקרו תהיה תמיד לפני הקריאה למאקרו  • נדרש שהקדם-אסמבלר ייצור קובץ עם הקוד המורחב הכולל פרישה של המאקרו (הרחבה של קובץ המקור המתואר בהמשך).  "קובץ המקור המורחב" הוא "קובץ מקור" לאחר פרישת המאקרו, לעומת "קובץ מקור ראשוני" שהוא קובץ הקלט למערכת, כולל הגדרת המאקרואים.  משפטים: קובץ מקור בשפת אסמבלי מורכב משורות המכילות משפטים של השפה, כאשר כל משפט מופיע בשורה נפרדת. כלומר, ההפרדה בין משפט למשפט בקובץ המקור הינה באמצעות התו ‘n) ‘\שורה חדשה).  אורכה שורה בקובץ המקור:  80 תווים לכל היותר (לא כולל התו n\).  4 סוגי משפטים:    משפט הנחיה: תווית (אופציונלי), שם ההנחיה (מתחיל ב-. וכולו אותיות קטנות), פרמטרים. ללא ARE, תופס את כל 12 הסיביות של המילה.   1. .data - מספרים שלמים חוקיים מופרדים ע"י פסיק. רווחים וטאבים חופשי, פסיק אחד בודד בין כל פרמטר בלבד. ללא פסיק לפני או אחרי אוסף הפרמטרים. משמש להקצאת תאי זיכרון לאחסון הנתונים, לאחריו מתבצע קידום מונה הנתנונים בהתאם למספר הערכים. אם ישנה תווית אז היא נכנסת אל טבלת הסמלים עם ערך מונה הנתונים לפני הקידום (וזהו בעצם מיקום הנתון הראשון בסט שהתווסף). 2. .string - פרמטר בודד - מחרוזת ascii עם מרכאות. כל תו הופך למילה בפני עצמה. בסוף מתווסף הערך 0 (0\) שמסמן את סוף המחרוזת. תווית - בדומה למה שקורה בדאטה, מסמלת את כתובת תחילת המחרוזת. 3. .entry - פרמטר אחד - שם תווית שמקבלת את ערכה בקובץ הנוכחי ומאפשרת לקבצים אחרים להשתמש בה כאופרנד של הוראה. לא ניתן להצמיד לפני כן תווית למיקום ההנחיה הזו. נוציא הודעת אזהרה - האסמבלר יתעלם מתווית זו. 4. .extern - פרמטר אחד - שם של תווית שמוגדרת בקובץ אחר ונרצה להשתמש בה פה. הכיוון ההפוך של entry. לא ניתן להגדיר את 2 ההנחיות האלה על אותה תווית באותו קובץ. לא ניתן להצמיד לפני כן תווית למיקום ההנחיה הזו. נוציא הודעת אזהרה - האסמבלר יתעלם מתווית זו.   משפט הוראה  מורכב מ:   1. תווית (אופציונלי) - אם קיימת תוכנס לטבלת הסמלים וערכה יהיה מען המילה הראשונה של ההוראה בתוך הקוד שבונה האסמבלר 2. שם הפעולה 3. אופרנדים אם אמורים להיות.   שם הפעולה תמיד באותיות קטנות, והוא אחת מ- 16 הפעולות שצויינו לעיל.  לאחר שם הפעולה עם לפחות רווח/טאב אחד יופיעו האופרנדים אם יש. רווחים וטאבים חופשי, פסיק אחד בודד בין כל פרמטר בלבד. |
| **איפיון שדות במשפטים של שפת האסמבלי** | תווית: סמל שמוגדר בתחילת משפט הוראה או הנחיית data/string. מתחילה באות אלפביתית, אח"כ אותיות אלפבתיות קטנות או גדולות ו/או מספרים. אורך מקסימלי - 31  סוף הגדרת שם תווית: ":" לא נכלל בתווית עצמה, חייב להיות צמוד ללא רווח.  אסור שתווית תוגדר יותר מפעם אחת. אותיות קטנות וגדולות נחשבות שונות. מילים שמורות של השפה אסורות לשימוש. ערך התווית - תלוי בהקשר (תווית הנחיה תקבל ערך DC נוכחי, תווית הוראה תקבל ערך IC נוכחי).  מספר: יכול להתחיל בסימן +/-, לאחר מכן סדרה של מספרים בבסיס עשרוני. אין תמיכה בבסיסים אחרים, אין תמיכה בלא שלמים. ללא פריפיקס.  מחרוזת: סדרת תווי ascii מוקפת במרכאות כפולות (ואינן חלק מהמחרוזת). |
| **קדם אסמבלר** | לפני שהאסמבלר מתחיל לעבד את הקוד, שלב הקדם-אסמבלר מבצע פרישת מקרואים. בכל מקום בו מאוזכר שם של מאקרו שהוגדר בקובץ קודם לכן תתבצע הצבה של קטע הקוד שהמאקרו מבצע. לבסוף יוצא קובץ am. אם אין בכלל מאקרואים אז קובץ הam יהיה זהה לקובץ המקור (as).  אלגוריתם מוצע לקדם אסמבלר:  1. קרא את השורה הבאה מקובץ המקור. אם נגמר הקובץ עבור ל9 (סיום)  2. האם השדה הראשון הוא שם מאקרו המופיע בטבלת המאקרו (כגון m1)? אם כן, החלף את שם המאקרו והעתק במקומו את כל השורות המתאימות מהטבלה לקובץ, חזור ל 1. אחרת, המשך.  3. האם השדה הראשון הוא "mcro"? אם לא, עבור ל-6.  4. הדלק דגל "יש מאקרו"  5. הכנס לטבלת שורות מאקרו את שם המאקרו.  6.קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עבור ל-9. אם דגל "יש מאקרו" דולק ולא זוהתה תווית endmcro הכנס את השורה לטבלת המאקרו ומחק את השורה הנ"ל מהקובץ. אחרת, (לא מאקרו) חזור ל 1.  7. האם זוהתה תווית endmcro? אם כן, מחק את התווית מהקובץ והמשך. אם לא, חזור ל-6.  8. כבה דגל "יש mcro". חזור ל-1.  9. סיום: שמירת קובץ מקרו פרוש. |
| **אסמבלר עם 2 מעברים** | **מעבר ראשון:**  זיהוי סמלים (תוויות), בניית טבלת ערכים ונתינת ערך מספרי דצימלי לסמל (המען בזיכרון שהסמל מייצג), לאחר מכן הצבת הערך המספרי בקוד בכל מקום בו הסמל מופיע כאופרנד.  נבצע ספירה של המקומות בזיכרון שההוראות תופסות (כל הוראה נטענת למקום העוקב להוראה הקודמת). הספירה תציין את מען ההוראה הבאה. הספירה תיעשה בשלב זה ותאוחסן במונה ההוראות IC.  ערך התחלתי: 0, מיקום בפועל בקוד: 0 + 100 (100 הוגדר להיות המקום ההתחלתי בזיכרון לכל התוכנית). ה-IC מתעדכן בכל שורת הוראה המקצה מקום בזיכרון. לאחר שהאסמבלר קובע מהו אורך ההוראה, ה-IC מוגדל במספר התאים (מילים) הנתפסים על ידי ההוראה, וכך הוא מצביע על התא הפנוי הבא.  טיפ מוסווה: שימוש בenums לאחסון ושליפה מהירה של קידודי שמות פעולה. בחוברת כתוב "טבלה" אבל ברור שזו הכוונה. |
| **הפרדת הוראות ונתונים** | חייבת להתבצע הפרדה בין ההוראות והנתונים בקוד המכונה (אין הכרח בקוד המקור). איך נבצע? IC סופר ומארגן את שיבוץ ההוראות, DC סופר ומארגן את שיבוץ הנתונים. כשנגיע לשלב ההמרה נשבץ תחילה את ההוראות בעזרת הIC + INITIAL\_VAL ולאחר מכן נשבץ את הנתונים בעזרת הDC + IC + INITIAL\_VAL. |
| **גילוי שגיאות בתוכנית המקור** | הנחת יסוד: אין שגיאות במאקרו, לכן בקדם אסמבלר אין שלב גילוי שגיאות.  לעומת זאת, באסמבלר נכסה שגיאות תחביר של תוכנית המקור (פעולה שלא קיימת, מספר אופרנדים שגוי, סוג אופרד שלא מתאים לפעולה, שם אוגר לא תקין וכו'). כמו כן - ווידוי שכל סמל מוגדר פעם אחת בלבד.  בדיקת השגיאות אמורה להתבצע בעיקר על שורה. אם יש שגיאה בקוד שמוגדר במקרו השגיאה עלולה להיות מוכפלת ולא נתן בשלב זה כבר לחסוך גילויי שגיאה כפולים.  הודעת שגיאה תודפס לstdout (פלט סטנדרטי). יצויין מספר השורה בקובץ המקור (השורות מתחילות מ-1).  גם אם התגלתה תקלה לא עוצרים בזיהוי התקלות, ממשיכים לרוץ על בדיקת התוכנית. אם יש שגיאות לא מייצרים את קבצי הפלט. שיטות מיעון חוקיות לאופרנד: |
| **אלגוריתם שלדי** | קוד המכונה מתחלק ל"תמונת ההוראות" (הקוד) ו"תמונת הנתונים" (הדאטה). לראשון יש מונה IC, לשני יש מונה DC. נסמן ב-L את מספר המילים שתופס קוד מכונה של הוראה נתונה. בכל מעבר נתחיל לקרוא את הקובץ מההתחלה.  **מעבר ראשון:**  1. אתחל DC=0, IC=0.  2. קרא את השורה הבאה מקובץ המקור. אם נגמר קובץ המקור, עבור ל16.  3. האם השדה הראשון הוא סמל? אם לא, עבור ל-5.  4. הדלק דגל "יש הגדרת סמל"  5. האם זוהי הנחיה לאחסון נתונים, כלומר, האם הנחית data. או string. ? אם לא, עבור ל-8.  6. אם יש הגדרת סמל (תווית), הכנס אותו לטבלת הסמלים עם סימון (סמל מסוג data) - ערכו יהיה DC. (אם הסמל כבר נמצא בטבלה, יש להודיע על שגיאה).  7. זהה את סוג הנתונים, קודד אותם בזיכרון, עדכן את מונה הנתונים DC בהתאם לאורכם, חזור ל-2.  8. האם זו הנחיית extern. או הנחיית entry. ? אם לא, עבור ל11.  9. האם זוהי הנחיית extern. ? אם כן, הכנס כל סמל (אחד או יותר) המופיע כאופרנד של ההנחיה לתוך טבלת הסמלים ללא ערך, עם סימון (סמל מסוג external).  10. חזור ל-2.  11. אם יש הגדרת סמל, הכנס אותו לטבלת הסמלים עם סימון (סמל מסוג code). ערכו יהיה IC (אם הסמל כבר נמצא בטבלה יש להודיע על שגיאה).  12. חפש את שם הפעולה בטבלת שמות הפעולות, ואם לא נמצא – הודע על שגיאה בשם ההוראה.  13. נתח את מבנה האופרנדים של ההוראה וחשב את L. בנה כעת את הקוד הבינארי של המילה הראשונה של הפקודה.  14. עדכן IC = L + IC  15. חזור ל-2. |
| **קבצי קלט ופלט של האסמבלר** | בהפעלת האסמבלר מעבירים באמצעות ארגומנטים של שורת הפקודה רשימה של קבצי מקור (אחד או יותר) שהסיומת שלהם היא as, אבל בכתיבה שלהם כארגומנט המשתמש נדרש לרשום את שמם ללא הסיומת. אלו קבצי טקסט עם תוכניות בתחביר של שפת האסמבלי שלנו.  האסמבלר פועל על כל קובץ בנפרד ויוצר עבורו את קבצי הפלט:   1. קובץ am - קובץ המקור לאחר שלב הקדם-אסמבלר (פרישת המקרואים). 2. קובץ ob - קוד המכונה הסופי |
| **אופן פעולת האסמבלר** | האסמבלר מחזיק שני מערכים - מערך ההוראות ומערך הנתונים. מערכים אלו נותנים למעשה תמונה של זיכרון המכונה (גודל כל כניסה במערך זהה לגודלה של מילת מכונה: 12 סיביות).  במערך ההוראות מכניס האסמבלר את הקידוד של הוראות המכונה. במערך הנתונים מכניס האסמבלר את קידוד הנתונים (שורות מסוג data. string.).  לאסמבלר יש שני מונים: מונה ההוראות (IC) ומונה הנתונים (DC). מונים אלו מצביעים על המקום הבא הפנוי במערכים לעיל. כשמתחיל האסמבלר לעבור על קובץ מקור, שני מונים אלו מאופסים.  בנוסף יש לאסמבלר טבלה, לכל התוויות בהן נתקל האסמבלר במהלך המעבר על הקובץ - טבלת סמלים (table-symbol). לכל סמל (תווית) נשמרים שמו, ערכו וטיפוסו (external או relocatable).  האסמבלר קורא את קובץ המקור שורה אחר שורה, מחליט מהו סוג השורה (הערה, הוראה, הנחיה או שורה ריקה) ופועל בהתאם.   1. שורה ריקה/הערה – מתעלם וממשיך הלאה 2. שורת הוראה - מזהה את הפעולה ואת שיטות המיעון של האופרנדים, וקובע את ערך האופרנדים כך:    * אם האופרנד הוא רגיסטר – מספר הרגיסטר    * תווית (מיעון ישיר) – ערך התווית כמו שמופיע בטבלת הסמלים    * מספר (מיעון מיידי) - המספר עצמו   האופרנד מזוהה לפי התחביר שלו.  בשלב הבא האסמבלר מכניס למערך ההוראות במקום עליו מצביע הIC את קוד המילה הראשונה (לאחר קידוד). מילה זו מכילה את קוד הפעולה, ואת שיטות המיעון.  לאחר מכן משריין האסמבלר מקום במערך עבור המילים הנוספות הנדרשות עבור ההוראה, אם נדרשות כאלה, ומגדיל את מונה ההוראות בהתאם. אם אלו מילים שניתן לקודד מיד (כלומר לא תווית) אז הוא כבר מקודד ומכניס אותם למערך.  אם בשורת ההוראה קיימת תווית, אזי התווית מוכנסת אל טבלת הסמלים תחת השם המתאים,  ערך התווית הוא ערך מונה ההוראות לפני קידוד ההוראה, וסוג התווית הוא relocatable.   1. שורת הנחיה –  * Data - האסמבלר קורא את רשימת המספרים, מכניס אותם אחד אחד אל מערך הנתונים, ומקדם את DC באחד עבור כל מספר שהוכנס. אם יש תווית היא מוכנסת לטבלת הסמלים ומקבלת את ערכו של DC לפני קידום. הטיפוס של התווית הוא R ומסומן שהוא הוגדר בחלק הנתונים. בסוף המעבר הראשון ערך התווית יעודכן בטבלת הסמלים ע"י הוספת ערך IC לערכו (ואז מתקבל המיקום הסופי האמיתי של התווית). * String – כמו הדאטה, רק שכאן קודי הascii של המחרוזת מוכנסים אל מערך הנתונים כל אחד בנפרד + הערך "/0" בסוף. מונה הנתונים מקודם במספר הערכים שהוכנסו, כלומר אורך המחרוזת + 1 (עבור /0). תווית – בדומה לדאטה. * Entry – האסמבלר רושם את הבקשה להכניס את התווית המוכרזת אל קובץ הent. האסמבלר רושם את הבקשה ובסיום העבודה התווית תירשם בקובץ הent. * Extern – הצהרה על סמל חיצוני שהקוד בקובץ המקור עושה בו שימוש. האסמבלר מכניס את הסמל לטבלת הסמלים. ערכו הוא 0 וטיפוסו E. לא ידוע איפה הוא מוגדר במקור וזה גם לא משנה לאסמבלר.  אפשר להשתמש בשם של סמל גם אם ההצהרה עליו מגיעה רק בהמשך הקובץ (גם אם מדובר בסמל חיצוני). |
| **פורמט קובץ ה-object** | קידוד ההוראה הראשונה יכנס למען 100 (בבסיס 10) בזיכרון, קידוד ההוראה השנייה יכנס למען העוקב (תלוי במספר המילים של ההוראה הראשונה), וכך הלאה עד להוראה האחרונה.  מיד לאחר קידוד ההוראה האחרונה, מכניסים לתמונת הזיכרון את קידוד הנתונים (‘string.’ data. ‘). הנתונים יוכנסו בסדר בו הם מופיעים בקובץ המקור. אופרנד של הוראה שמתייחס לסמל שהוגדר באותו קובץ, יקודד כך שיצביע על המקום המתאים בתמונת  הזיכרון שבונה האסמבלר.  נשים לב שהמשתנים מופיעים בתמונת הזיכרון אחרי ההוראות. זוהי הסיבה בגללה יש לעדכן בטבלת הסמלים, בסוף המעבר הראשון, את ערכי הסמלים המגדירים נתונים (סמלים מסוג  data).  עקרונית, קובץ object מכיל את תמונת הזיכרון שתוארה כאן. קובץ object מורכב משורות של  טקסט כדלקמן:  השורה הראשונה היא כותרת המכילה שני מספרים בבסיס 10: האורך הכולל של קטע ההוראות (במילות זיכרון) ואחריו האורך הכולל של קטע הנתונים (במילות זיכרון). בין שני המספרים יש רווח אחד.  השורות הבאות מכילות את תוכן הזיכרון. בכל שורה מופע תוכן של מילה אחת, לפי הסדר החל מהמילה בכתובת 100. תוכן המילה מקודד בשיטת 64Base.  <https://en.wikipedia.org/wiki/Base64>  מילה היא בגודל 12 ביטים. כל 6 ביטים יומרו לתו מתאים לפי הטבלה המגדירה את base 64. לכן בכל השורות מלבד השורה הראשונה יהיו בדיוק 2 תווים בקוד base64. |
| **פורמט קובץ ה-entries** | כל שורה מכילה שם של סמל שהוגדר כ- entry ואת ערכו כפי שנמצא בטבלת הסמלים. הערכים מיוצגים בבסיס 10. |
| **פורמט קובץ ה- externals** | כל שורה מכילה שם של סמל שהוגדר external,  וכתובת בקוד המכונה בה יש קידוד של אופרנד המתייחס לסמל זה. כמובן שייתכן ויש מספר כתובות בקוד המכונה בהם מתייחסים לאותו סמל חיצוני. לכל התייחסות כזו תהיה שורה נפרדת  בקובץ ה-externals. הכתובות מיוצגות בבסיס 10.  ייתכן ויש מספר כתובות בקוד המכונה בהן מילות-המידע מתייחסות לאותו סמל חיצוני. לכל כתובת כזו תהיה שורה נפרדת בקובץ ה-externals. |
| **סיכום והנחיות כלליות** |  |
| **חידודים מפורום הקורס (2023ב)** | **תוויות**   1. תווית יכולה להכיל בתוכה שם של פקודה כל עוד היא תת מחרוזת (לדוגמה h2mov). 2. יכולות להיות .data או .string ללא תווית. גם שם כמובן מעלים את המונה. 3. תווית לא יכולה להיות מוגדרת גם בקובץ וגם בחיצוני. 4. הגדרת תווית יכולה להתבצע פעם אחת בלבד (שם תווית ולאחר מכן נקודותיים). 5. תווית יכולה להיות מוגדרת ללא אפיון של entry או extern. תווית שמוגדרת כentry היא תווית ככל התויות אלא שניתן להשתמש בה בקבצים אחרים. 6. תווית שהוגדרה כ entry אך לא הוכרזה באופן עצמאי -> שגיאה, כמו תוויות שנקראו ולא הוכרזו קודם. entry לא מכריז על תווית אלא רק מציין שאפשר להשתמש בה אח"כ בקבצים אחרים. 7. אם קיימת הנחיית extern על תווית וגם הגדרה של אותה תווית באותו קובץ זו שגיאה. 8. מקרה של תווית ללא הנחיה או הוראה אחריה יש להודיע על שגיאה. 9. בתחילת ההוראה הראשונה בקובץ אין שום חיוב שתהיה תווית. 10. אם מתגלית שגיאה בשורה שיש בה תווית בתחילתה או שמשתמשת בתווית מוציאים הודעת שגיאה באשר לשגיאה הרלוונטית וממילא אין ליצור קבצי פלט. 11. תווית בשם של מאקרו – תהיה שגיאה ממילא בעיבוד ולכן אפשר לציין בהערה בתיעוד שממילא תתקבל שגיאה. בכל זאת אפשר לנסות להכניס את זה בהודעה למשתמש אם רוצים. 12. אם יש תווית שהוגדרה ואין בה שימוש – לא שגיאה. כנ"ל מקרו. 13. אם הייתה הגדרה של extern או entry שלא נעשה בהם שימוש - אחד המנחים ענה שזו שגיאה, אבל אסתר ענתה שזו לא שגיאה... יש הבדל בין תווית חיצונית לתווית פנימית:   תווית חיצונית: ניתן להצהיר (.extern LABEL) על תווית כתווית חיצונית ולא להשתמש בה בכלל בפקודות ההוראה. במקרה כזה אין להוסיף את התווית לקובץ ext. . אם אין תוויות חיצוניות נוספות אין ליצור קובץ ext . תווית המוצהרת כתווית חיצונית ובנוסף גם מוגדרת כתווית בקובץ זו שגיאה (.extern LABEL and LABEL: ).  תווית פנימית: חובה להגדיר תווית המוצהרת כתווית פנימית, אחרת - שגיאה. לכן, אם יש שורה כזו ".entry LABEL" ואין שורה כזו "LEBEL:" אזי יש לדווח על שגיאה. שימו לב שתוכן קבצי ext/ent מעט שונים.   1. הכרזה על תווית כ- entry. או extern. פעמיים – יש סתירה, כמה פעמים נכתב שיש להוציא שגיאה, כמה פעמים נכתב שזה בסדר. **אסתר ענתה כמה פעמים:** הנחיית entry/extern יכולה להתבצע כמה פעמים וזו לא שגיאה (.extern LABEL .extern LABEL). **זו המסקנה הסופית?** אפשר להצהיר כמה פעמים על תווית entry/extern. אפשר להוציא אזהרה.   **משפטי הנחיה**   1. הופעה של תווית במקום ערך מספרי ב.data למשל: .data VAL **- החוברת** הגדירה שלא ו**אסתר** גם אישרה שככה כתוב בחוברת אבל **רועי** קודם לכן אמר שבאסמבלר תקני יש שימוש במהלך כזה ולכן נראה לו שיש לאפשר זאת. **מאז ענו כמה פעמים שלא.** אין לאפשר תווית בהנחיית data. 2. בstring ניתן להצהיר רק על מחרוזת אחת בכל פעם. 3. אי אפשר להגדיר את אותה תווית גם כentry וגם extern. 4. אחרי ההנחיות extern ו entry חייב להתקבל שם של תווית. באשר לשאלה "האם ניתן לציין כמה תוויות יחד בהנחיות אלו" **נראה שיש סתירה.** בעמוד 29 בחוברת מצויין שextern מקבל כפרמטר רק תווית אחת, אבל בעמוד 36 באלגוריתם של המעבר הראשון נראה שיש תמיכה בהצהרה על מספר תוויות במכה אבל זה מפורש רק בextern. **רועי הבהיר בהמשך ש** **36 הוא הכלל לפיו צריך לנהוג. עמוד 29 היא הדגמה מקומית ולא ההוראה לביצוע, כלומר יכול להיות יותר מתווית אחת.** עמוד 36 נותן תשובה רק באשר לextern, מה לגבי entry? **אסתר** ענתה בכמה פעמים שלגבי entry יש הגבלה לתווית אחת, אך לextern אין הגבלה, וציינה שהתוויות יהיו מופרדות באמצעות פסיקים.  **חיים** ענה בתגובה יותר עדכנית שמותר רק תווית אחת בלבד ב-2 סוגי ההנחיות הנ"ל. **אסתר** ענתה לאחר מכן שיש להתייחס למופיע באלגוריתם השלדי, כלומר כאמור לextern מותר יותר מתווית אחת. **העדכון האחרון (מה06/08) של אסתר** הוא הבהרה שדווקא כן ניתן לכתוב מספר תוויות הן לאחר פקודת extern והן לאחר פקודת entry. ההפרדה בין התוויות תהיה על ידי פסיק. **מתבקש שתצא הודעה נעוצה בנושא זה שמציינת שכלל המנחים מעודכנים בנוגע להנחיה זו.**   מסקנה סופית – אפשר להצהיר על מספר תוויות לאחר הנחיית entry/extern   1. במשפט הנחייה - האם חייב להיות רווח/טאב בין תווית (אם יש) לבין הנקודה, בין הנקודה לשם ההנחיה, בין שם ההנחיה לפרמטרים ובמשפט הוראה האם חייב להיות רווח בין התווית (אם יש) לשם הפעולה? **רוני ענה שלא חייב, אבל אני חושב שזה לא תקין. בוודאי בין הנקודה לשם ההנחיה ובין ההנחיה לפרמטרים.** האם אפשר לחייב בכל זאת? חייב להיות תו לבן בין הנקודותיים של התווית לנקודה של ההנחיה. לא חובה שיהיה רווח בין התווית לשם הפעולה. יש לשים לב לתווים לבנים אופציונאלים ולפסיקים תקינים. 2. מחרוזת חוקית היא סדרת תווי ascii נראים (שניתנים להדפסה), המוקפים במרכאות כפולות. זה כולל תווים לבנים (רווח, טאב). תווים שאינם נראים הם תווים מיוחדים אחרים, אפשר לחפש בגוגל... חיים מאשר שתווים מותרים הם החל מהערך 32 בטבלה - רווח (כולל) עד הערך 126 - ~. 3. במחרוזת ריקה יש להכניס תו סיום מחרוזת בלבד. מה שקיים בתוך הגרשיים זה מה שיש להכניס למחרוזת. אם יש תו / יש להכניס אותו. אין צורך להמיר לescape characters. אם יש מרכאות באמצע הטקסט אבל יש גם מרכאות פותחות וסוגרות באופן תקין אפשר להחליט לכלול את המרכאות הפנימיות כחלק מהמחרוזת ואפשר לחילופין שלא אבל אז צריך לנהל את השגיאה. 4. במקרה של escape char במחרוזת יש להכניס את התו באותה צורה בה הוא הוזן (לדוגמה /t יוכנס כ"/t". 5. אורך שורה מקסימלי 80 (לא כולל התו של השורה החדשה), ז"א שלמחרוזת צריך להקצות 81 מקומות (בפועל הכוונה מן הסתם לעשות את זה כולל מילת ההנחיה של המחרוזת).   **הוראות**   1. מה קורה אם יש לי הוראה עם שני אופרנדים, כאשר אחד מהם תווית extern ואחד תווית רגילה. האם היא תקבל R או E?  **זו שגיאה מלכתחילה. אין פקודה שמקבלת 2 אופרנדים שהם תווית...**   **שורת הערה**   1. שורת הערה יכולה לבוא כשורה נפרדת בלבד. התו הראשון חייב להיות ; 2. שורה שיש בה רווחים וטאבים לפני ; היא עדיין שורת הערה (לדלג על התווים הלבנים) – יש סתירה! רוני בן ישי ענה בפורום שלא וגם אסתר ענתה שלא, אבל מנחה אחר ענה קודם שכן... **לעניות דעתי נראה שיש להתיר, כיוון שבשפות המצויות בשימוש נפוץ הדבר הזה מותר.**   לפי המופיע בחוברת המטלות - התו הראשון צריך להיות “;” על מנת להצהיר על שורה כשורת הערה  **מעברים וניהול נתונים**   1. עקרונית אפשר במעבר הראשון רק לשמור את הכתובות של הlabel בטבלת הסמלים מבלי לקודד את כל שאר הדברים שניתן לקודד ובמעבר השני לקודד פשוט הכל ביחד. מה שכתוב בחוברת לגבי זה זו המלצה. 2. אפשר להכניס במערך הנתונים ובמערך הפקודות ערכים בבסיס עשרוני (ורק כשנכתוב לקובץ נמיר לבינארי). 3. ניתן להחזיק את הנתונים בבסיס 64 כבר כשהם מקודדים ולא לשמור אותם בצורה בינארית, בתכלס זה לא ממש קריטי. 4. אפשר למחוק הערות ושורות ריקות כבר בשלב הקדם אסמבלר, אין שחובה לשמור אותם גם בקובץ המקור המורחב, אבל זה לבחירתנו בגדול. 5. טבלת אחסון מקרואים – לבנות מבנה זיכרון דינאמי 6. IC+DC צריך להיות קטן שווה ל924 (100 תאים ראשונים בזיכרון "שמורים"). 7. הזיכרון מנוהל עבור כל קובץ בנפרד. **לכל קובץ מקור יש לאתחל את כל המשתנים.  לכל קובץ יש מערך של נתונים ומערך של פקודות משלו. IC, DC מתאפסים. לכל קובץ מקור יש טבלת סמלים משלו.** 8. אין להכניס לטבלת הסמלים את הערך עצמו שהסמל מגדיר (רשימת המספרים או המחרוזת), הערכים נכנסים לטבלת הנתונים (data image). 9. מימוש הייצוג הבינארי כרצוננו, אפשר כמצביע למערך של char ואפשר כstruct וכו', כמובן עדיף לשמור על יעילות מירבית וצריך לשמור על גבולות. 10. בקידוד פקודה סדר הקידוד (המילים הנוספות) צריך להיות לפי סדר האופרנדים. 11. לפי האלגוריתם המוצג בחוברת יש טבלת סמלים אחת בלבד ולכל שורה של הטבלה מכניסים סימון דאטה או קוד בהתאם לסמל (כך לפי התשובה של אסתר), אבל אם אני זוכר נכון כתוב שהאלגוריתם לא מחייב... 12. התוכנית מקמפלת עבור 2'compliment machine. גם הקמפול עצמו יתבצע במכונה כזו (לפי אסתר). זה משנה כמובן עבור bitwise operations. 13. אין מניעה להשתמש במערכים כל עוד אין סתירה בין השימוש ובין הצורך על פי אפיון המטלה 14. הטווח של מספר כאופרנד בהוראה הוא 10 ביטים ולכן לו טווח אחד, ואילו עבור מספר בהנחיה הוא 12 ביטים ולכן לו טווח אחר. **לשים לב למגבלת הביטים השונה!** 15. ניתן כבר בשלב הקדם אסמבלר למחוק רווחים עודפים. 16. אין צורך להדפיס את טבלת המאקרו או את הטבלה של הסמלים. 17. אם מתקבל קובץ ריק אין צורך ליצור קבצי פלט 18. הדרישה של המטלה היא שלכל תא בזיכרון יהיה 12 BITS בדיוק, לא פחות ולא יותר. 19. אם יש שורה שארוכה מ80 תווים אפשר להתעלם מהשורה או להתייחס ל80 התווים הראשונים. כרצונך (אסתר). חשוב לזכור שבמקרה כזה אין פלטים מאחר ויש שגיאה בקובץ (שורה ארוכה מ80 תווים). 20. האם ספירת התווים המקסימליים בשורה לוקחת בחשבון תווים לבנים? הרי נאמר שצריך להתעלם מהם בהקשרים אחרים.   80 כולל תווים לבנים   1. אפשר להגדיר מספר חד ספרתי של משתנים גלובלים. מומלץ להימנע.   **ניהול שגיאות ואזהרות**   1. מספיק לגלות שגיאה אחת בשורה ולהמשיך לבאות, אין צורך להמשיך לגלות שגיאות בשורה הזו. 2. עבור שגיאה במעבר ראשון/שני קובץ הam כבר קיים ולכן נשאיר אותו כמו שהוא ואילו את הקבצים האחרים לא ניצור. 3. האם צריך לבדוק שאורך שורה לא גדול מ-80 בשלב הקדם אסמבלר או במעבר הראשון? **רועי:**  אם זה לא כתוב בצורה חחע במטלה אז אפשר כך או אחרת רק תעד את זה בקוד. לי נראה טבעי יותר לבדוק את זה לאחר פרישת המקרו.  **אסתר:**  נראה שצריך לבדוק זאת בשלב הקדם אסמבלר  (מה שאסתר אומרת נשמע הגיוני, את תוכן המאקרו ממילא בודקים ב-2 הוריאציות, וכל שנשאר הוא לראות אם שורת ההכרזה על המאקרו לא חורגת באורכה.) 4. הודעת שגיאה צריכה לציין את מספר שורת המקור שבה ישנה השגיאה. **לפי אסתר – הכוונה היא לקובץ הפרוש.** 5. אם מספר המתקבל כאופרנד או כנתון דאטה חורג ממגבלת הסיביות – להוציא שגיאה. 6. אפשר בנוסף להדפסות Error להוסיף גם הדפסות Info, לדוגמא - קובץ קיים, קובץ נוצר וכו'... 7. אם יש שגיאה בשם המאקרו, אין ליצור קבצי פלט. אך יש לגלות שגיאות נוספות (אם יש). 8. במקרה של שגיאה בהקצאת זיכרון יש להדפיס שגיאה ולצאת לגמרי מהתוכנית. 9. במקרים של אזהרות ללא שגיאות יש להוציא קבצי פלט. 10. הנחיית data או string ללא נתונים – **סתירה:** במקום אחד נאמר שיש להוציא אזהרה. **במקום אחר נאמר**: אם נמצאו משפטי הנחיה בלי ערך יש להודיע על שגיאה. כמו כן נאמר שאחרי ההנחיות extern ו entry חייב להתקבל שם של תווית. זו שגיאה 11. לאן להדפיס הודעת שגיאה? **בחוברת** כתוב ל- stdout. כך גם בפורום מספר פעמים. **אבל כתוב גם בפורום מספר פעמים** שצריך להוציא ל- stderr. **אסתר** ענתה stderr אבל גם אם נוציא לstdout לא ירד על זה ניקוד, ולאחר מכן **מנחה אחר** ענה בשרשור אחר stdout. יותר נכון להוציא שגיאות פלט לקובץ stderr. אך נתון לבחירתכם 12. צריך לבדוק שגיאות לוגיות (לפי אסתר – וגם לפי שאר המנחים..), לדוגמה: אם יש קוד כלשהוא אחרי הקריאה לפקודה stop זו שגיאה לוגית. 13. יש לגלות את כל השגיאות של שלב קדם אסמבלר אך אין לעבור לשלבים הבאים. 14. במצב בו מנסים ליצור קבצי .ob, ent ו-ext, והבקשה נכשלת אין לנסות להמשיך להפיק את הקבצים שכן מתאפשר, יש להמשיך לעיבוד של קובץ המקור הבא. מן הסתם להוציא כאן הודעת שגיאה... 15. אי אפשר להגדיר 2 מאקרואים באותו שם. זו שגיאה. אם יש דריסה של מאקרו , כלומר יש הגדרה חדשה של מאקרו שכבר הוגדר אך עם תוכן שונה יש להודיע על שגיאה, להשלים את מעבר הקדם אסמבלר על הקובץ לזיהוי שגיאות נוספות ולצאת. 16. אם קראתי לקובץ שלא קיים יש להודיע על כך ולהמשיך לקובץ הבא. 17. גם אם גילינו שגיאה במעבר הראשון יש להמשיך ולגלות את השגיאות של המעבר השני. 18. עבור חריגת זיכרון מוציאים שגיאה ומתייחסים כמו כל שגיאה אחרת.   **מאקרו**   1. המילים השמורות הן case sensitive ככה שאפשר להשתמש במילים האלה למקרואים או תוויות בעיוות. 2. שם של מאקרו צריך לעמוד בתנאים של שם תווית חוקית 3. קריאה למאקרו לפני הצהרתו – אפשר לעצור ולעבור לקובץ הבא (לפי רועי רחמני). **לפי מיכל ניתן להניח שזה לא קורה**. אפשר לציין בהערה את מה שמיכל ציינה. 4. האם קריאה למאקרו יכולה להופיע בשורה של LABEL? L1: m1? **אסתר** ענתה שכן, צריך לפרוש מאקרו בכל מקום בו הוא נמצא. **במקומות אחרים ענתה** שאפשר להניח ששם המאקרו צריך להופיע כמילה ראשונה בשורה (תווים לבנים אפשריים לפני). **האם הכוונה שהנושא לבחירתנו או שישנה הכרעה כלשהי בנושא? לבחירתכם. יותר קל להניח ששם מאקרו יחיד בשורה. פרישה בכל מקום תגרור שאלות נוספות (מופיעות בפורום) וכנראה גם שגיאות במעבר הראשון.** 5. לא ניתן להגדיר מאקרו בשורה עם תווים אחרים, פרט לתווים לבנים. 6. אין חיוב להזחה בשורות שבתוך המאקרו. זה שקוף. 7. אם העברנו קובץ באסמבלר בהצלחה ואז שינינו אותו ונוצרה שגיאה – האם בפעם הבאה שהקובץ נכנס לאסמבלר יש צורך למחוק את הקבצים מההרצה הקודמת? לכאורה הם כבר לא בתוקף. **אסתר:** השאלה לא ברורה. כל הרצה בפני עצמה - אם התרגום עובר בהצלחה נוצרים הקבצים. אפשר לשנות את הקובץ (אם כי לא מומלץ) אך השינוי לא אמור להוביל לשינוי של הרצות קודמות. התשובה היא שלא צריך להתייחס לקבצים קודמים אם נוצרו בהרצות קודמות. יש להקפיד על פלט השגיאות.. 8. אם הוגדר מקרו עם שם ולאחריו כמובן הופיע endmcro אבל ביניהם לא היה שום טקסט, כלומר, המקרו ריק, אז אפשר פשוט להמשיך כרגיל. 9. אם יש מקרו שהוגדר ואין בו שימוש – לא שגיאה. כנ"ל תווית. 10. ניתן להניח שגם אין **שימוש** של מאקרו אחד בתוך מאקרו אחר |
| **הבהרות סופיות של מיכל הרכזת­** | **סיכום לגבי המאקרו:**  נושאים שניתן להניח ולא צריך לבדוק:  1) הגדרת מאקרו תבוא לפני התיחסות למאקרו  2) בכל הגדרת מאקרו יופיע בסוף סיומת המאקרו (endmcro)  3) אין שגיאות בגוף המאקרו (את השגיאות, אם יש, מגלים בשלבים הבאים)  נושאים שיש לבדוק:  1) שם המאקרו תקין (אינו שם הוראה וכדו..)  2) בשורת ההגדרה ובשורת הסיום אין תוים נוספים  אם יש שגיאה בשלב פרישת המאקרו - לא ניתן ליצר קובץ מקור מורחב. כתוצאה מכך, אי אפשר לעבור לשלבים הבאים ויש לסיים את התוכנית ולעבור לקובץ המקור הבא (אם קיים). |

בהצלחה!

הערות לעצמי:

ירוק - קראתי וטיפלתי

סגול - דילגתי

אדום – צריך לחזור שוב, לא מבין