**מילים קשות:**  
אוגר – register   
סיבית – bit   
מילה (מילת זיכרון)/ תא – byte   
המשלים ל 2 - 2’s complement arithmetic  
מען – כתובת   
מיעון – גישה לתא זיכרון כדיי לאחסן בו/להוציא ממנו נתונים  
PC - Program counter. This counter always points to the next instruction line, to be fetched into the CPU.

**נתוני המחשב הדימיוני:**   
\* כל מילת זיכרון היא 12 סיביות – Each byte is 12 bit  
\* המחשב עובד רק עם מספרים שלמים (שליליים וחיוביים)  
\* כל פעולה ארית'מטית משתמשת בשיטת המשלים ל2.  
\* למחשב יש 8 אוגרים(registers) רגילים. שמותיהם r0, r1…. r6, r7. כל רגיסטר בגודל של 12 סיביות.  
\* יש עוד אוגר בשם PSW. האוגר הזה מכיל מספר דגלים (so it will be a structure)  
\* למחשב גודל זיכרון של 1024 תאים (1024 bytes = 1024\*12 bits)  
\* המחשב משתמש בascii בלבד.  
  
  
**נתונים אסמבלר:**  
\* IC & DC = 0 בהתחלה  
\* כאשר האסמבלר נתקע בתווית בתחילת הוראה, הוא מוסיף אותה לטבלה ונותן לה את ערך הIC הנוכחי.  
\* צריך גם סופר שורות, בשביל לציין איפה היית שגיאה  
\* כאשר האסמבלר מצא שגיאה, יש להמשיך בקריאת הקוד כדיי למצוא שגיאות נוספות, אך לא ליצור קבצי פלט  
\* משתנה L – יחזיק כמה מילות\תאי זיכרון השורה הספציפית לוקחת כדיי שנדע בכמה להעלות את IC& DC

**הוראת מכונה :**  
**גודל ההוראה בזיכרון (תאורטי)**\* כל הוראת מכונה לוקחת 1-3 תאי זיכרון. תלוי בפעולה שרוצים לעשות.  
 - הוראה חסרת אופרנדים, תיקח מילה/תא זיכרון אחד   
 - הוראה בעלת אופרנד אחד תיקח 2 מילים/תאים. התא הראשון יוקדש למבנה ההוראה הכתוב למטה והתא השני  
 לציון האופרנד  
 - הוראה בעלת 2 אופרנדים תיקח 2-3 תאי זיכרון. התא הראשון יוקדש למבנה ההוראה, התא השני או שיתחלק בין שני   
 האופרנדים, או שכל אופרנד יקבל תא משלו. תלוי בסוג האופרנדים.  
 אם כל אופרנד לוקח תא זיכרון משלו (הוראה לוקחת 3 מילים בכללי), הראשון יהיה אופרנד המקור והשני יהיה  
 אופרנד היעד  
 **מבנה התא הראשון בהוראה**  
\* מבנה המילה/תא הראשון בהוראה תמיד זהה  
Table

Description automatically generated  
 0-1) סוג הקידוד   
 - מוחלט Absolute 00   
 - חיצוני External 01  
 - מיקום מחדש Relocatable 10  
 2-4) מייצג בבינארי את סוג אופרנד היעד (destination operand type. Can be value, variable, register)  
 5-8) מספר המייצג את הפעולה שאנו רוצים לעשות.  
 - יש 16 פעולות אפשריות. לכן המספר יהיה 0-15.   
 - שם הפעולה תמיד יהיה באותיות קטנות (יש לבדוק אם שם הפעולה קיים בטבלה: פעולה-מספר)  
 - לפי הפקודה נדע אם בכלל צריך את   
 9-11) מייצג בבינארי את סוג אופרנד המקור (source operand type. Can be value, variable, register)  
  
 **ARE**  
 \* כל מילת הוראה (לא משנה אם ראשונה שניה או שלישית) תכיל 2 סיביות שמייצגות את ה ARE  
 - מילה מייצגת משהו שמוגדר בקובץ אחר (סמל שהוגדר באמצעות .extern) ה 2 סיביות יהיו 01 (השאר יהיו 0)  
 - אם המילה מייצגת משהו שהוגדר בקובץ הנוכחי (סמל שמוגדר בקובץ זה) ה2 סביות יהיו 10  
 - אם המילה מייצגת כל דבר אחר (מספר, אוגר, פקודה) ה2 סביות יהיו 00  
  
  
**שיטת מיעון (עמ' 21)**  
\* שיטת המיעון מסבירה לנו כיצד כל סוג של אופרטור יוצג בתא הראשון של ההוראה וכיצד הוא הוא יאוכסן בבניארי   
 - (מיעון מידי) אופרנד מספר שלם – כאשר האופרנד הוא מספר שלם, הוא יוצג כ 001 באופרנד המתאים (מקור\יעד).  
 בזכרון הוא יתפוס שורה חדשה שמבנה: סיביות 0-1 יהיו ARE, סביות 2-11 יהיו היצוג הבינארי של המספר.  
 - (מיעון ישיר) אופרנד סמל/תווית – כאשר האופרנד הוא סמל שנוצר על ידי .data, .text או בתחילת הוראה  
 הוא יוצג כ 011 באופרנד המתאים (מקור\יעד).   
 בזכרון הוא יתפוס שורה חדשה שמבנה: סיביות 0-1 יהיו ARE, סביות 2-11 יהיו היצוג הבינארי של שורת הזיכרון שהוא  
 מייצג.  
 - (מיעון אוגר) אופרנד הוא אוגר – כאשר האופרנד הוא אוגר (אחד מה@r), הוא יוצג בתור 110 באופרנד המתאים.  
 כאשר יש אופרנד יעד שהוא אוגר, תווסף מילה\תא חדש שמבנה: סיביות 0-1 ARE ו 2-6 מספרו של האוגר בבינארי.  
 אם אופרנד המקור הוא אוגר, תווסף מילה\תא חדש שמבנה: סיביות 0-1 ARE ו 7-11 מספרו של האוגר בבינארי.  
 אם שני האופרנדים הם אוגרים, הם יחלקו מילה אחת 0-1 ARE, 2-6 יעד, 7-11 מקור.

**סוגי פקודות (עמ' 22-23)**  
\* פקודה הלוקחת שני אופרנדים - בתא ההוראה הראשון ישומשו כל הסיביות מ2-11 בוודאות.  
\* פקודה הלוקחת אופרנד אחד - בתא ההוראה הראשון ישומשו סיביות 5-8 ו 9-11 בוודאות.  
\* פקודה ללא אופרנדים – בתא ההוראה הראשון רק 5-8 ישומשו בוודאות.

**משפט הנחיה**:  
\* למשפטי הנחיה לא מצורפים סיביות ARE, והמשפט ימלא לבד את כל 12 הסיביות בתא הזכרון.  
\* תבנית משפט הנחיה  
 - בתחילתו יכול להופיע סמל ואחר כך רווח.  
 - תופיע נקודה ‘.’ וצמוד לנקודה ללא רווח יופיע שם ההנחיה. באותיות קטנות בלבד.  
 - לאחר שם ההנחיה יופיעו פרמטרים.  
**סוגי משפטי הנחיה**  
\* .data   
 - הפרמטרים של .data הם רשימת מספרים חוקיים (מספר אחד או יותר) המופרדים על ידי ','  
 - בין המספר לפסיק יכול להיות כל כמות של \s \t אבל בין כל שני מספרים חייב להיות פסיק אחד.  
 דוגמא: FuN12: -2, 14 , -52, 17  
 - .data מנחה את האסמבלר לדחוף את המספרים הללו במבנה הנתונים (לא ההוראות) ולהזיז את הDC   
 בהתאם לכמות המספרים שהכנסנו.  
 - אם המשפט מתחיל בסמל, הסמל יקבל את ערך הDC לפני הכנסת המספרים והזזתו. בעזרת הסמל נוכל לפנות  
 להתחלת הarray. בסוף המעבר הראשון, ערך הסמל אתעדכן ל: סמל += IC   
 Graphical user interface, text, application

Description automatically generated  
  
\* .string   
 - הפרמטר של .string הוא מחרוזת אחת של ascii.   
 - ערכם המספרי של תווי המחרוזת יוכנסו לתוך מבנה הנתונים, כל תו במילה אחרת. בסוף המחרוזת יוכנס גם הערך 0  
 בכדי לסמן שנגמרה המחרוזת.  
 - הDC יוגדל בהתאם לאורך המחרוזת + 1 (בגלל ה0).  
 - אם המשפט התחיל בסמל, הסמל יהיה שווה לDC לפני הכנסת התווים (כך שזה בעצם מיקום התוו הראשון).  
 בסוף המעבר הראשון, ערך הסמל אתעדכן ל: סמל += IC  
 - מחרוזת חוקית מוקפת במרכאות שאין נחשבות לחלק מהמחרוזרת “hello”  
\* .entry  
 - הפרמטר של .entry הוא סמל.   
 - ה .entry מצהיר על סמל זה כך שגם קבצי קלט אחרים יוכלו להשתמש בו.  
 ערך הסמל יהיה היכן שהוא מוגדר לראשונה בקובץ המקור הנוכחי.  
 - אם מופיע סמל לפני ה .entry יש להוציא שגיאה  
 - סמל שהוא פרמטר של .entry יתווסף לקובץ הפלט .ent  
  
\* .extern   
 - הפרמטר של .extern הוא סמל.   
 - ה . extern מצהיר על שאנו רוצים להשתמש בסמל שמוגדר בקובץ אחר.  
 - אם מופיע סמל לפני ה .extern יש להוציא שגיאה  
 - סמל שהוא פרמטר של .ext יתווסף לקובץ הפלט .extern שזה בתכלס כל מה שאנחנו צריכים לעשות אם   
 הנחיה מסוג זה

**משפט הוראה**:  
\*מבנה המשפט – Src operand(Optional) Dst opernad(optinal) Label(Optional) Command  
\* שם הפעולה יהיה תמיד באותיות קטנות  
\* במקרה שיש 2 אופרנדים יהיה פסיק אם כמות לא ידוע של \s \t בין האופרנד לבין הפסיק  
\* בין הפקודה לבין האופרנד הראשון יש כמות לא ידועה של \s \t  
\* אם יש סמל בתחילת ההוראה, הסמל יכנס לטבלת הסמלים ויקבל את מיקום המילה הראשונה של ההוראה בתמונת הזכרון.  
  
  
  
**תווית**  
**חוקי תווית**  
\* תתחיל באות אלפבטית (קטנה או גדולה)  
\* אורכה המקסימלי יהיה 31 תווים  
\* התוויות תחיל רק אותיות ומספרים   
\* אסור שתווית אחת תוגדר יותר מפעם אחת  
\* תווית חייבת להיגמר ‘:’   
\* בין הנקודותיים לפקודה כמות בלתי מוגדרת של \s \t  
\* שם של פעולה, הנחיה, אוגר לא יכולות להיות שמות של תוויות  
ערך התוויות   
\* תווית שהוגדרה ב .data או .string תקבל את ערך הDC הנוכחי  
\* תווית שהוגדרה ב .entry תקבל את ערך הIC של איפה שאותחלה פעם ראשונה ((Initilized ולא איפה שהוצהרה לראשונה.  
\* תווית שהוגדרה בשורת הוראה, תקבל את ערך הIC הנוחכי.  
  
  
**מספר**  
\* מספר יכול להיות רק שלם ובבסיס 10.  
\* מספר יכול להתחיל ב + - או בכלום.

**קובץ הקלט**  
**נתונים יבשים**  
\* מורכב מ 1-4 חלקים. Src operand(Optional) Dst opernad(optinal) Label(Optional) Command   
\* מקסימום 80 תווים (לא כולל ה "\n" ) אלה אם כן זאת שורת הערה (Comment)

**סוגי שורות**   
\* שורה ריקה - תחיל רק \s \t \n  
\* משפט הערה – יתחיל ב; צריך ישר לדלג על שורה מסוג זה  
\*משפט הנחיה - ישנם מספר סוגים של משפטי הנחיה. לרוב משפטי הנחיה יקצו ויאתחלו משתנים וערכים.  
\* משפט הוראה – מתואר לעומק למעלה (הוראות מכונה).

A screenshot of a computer

Description automatically generated with low confidence