מעבדה 2 -חלק תיאורטי

1. מחסנית היא מבנה נתונים שמשמש לארגון מידע כך שהאלמנט האחרון שנכנס הוא הראשון שיוצא (Last In, First Out - LIFO). שימוש נפוץ במחסנית הוא על מנת לעקוב אחר פונקציות בתוכנה, כך שכשאר פונקציה נקראת, הכתובת ממנה ה – PC צריך להמשיך לאחר סיום ביצוע הפונקציה נשמרת, והתוכנית ממשיכה באופן תקין משורת הקוד הבאה אחרי שורת הקריאה לפונקציה. בנוסף, מחסנית מאפשרת תמיכה בקריאות רקורסיביות בצורה טובה, ומאפשרת שימוש ברגיסטרים גם כאשר נגמרים הרגיסטרים לשימוש על ידי דחיפת ערכי הרגיסטרים למחסנית ושליפתם לאחר מכן.
כדי להשתמש במחסנית בשפת אסמבלי נשתמש בפקודות הבאות:
   - Pop- שליפת האלמנט מראש המחסנית והסרתו ממנה.
   - Push- דחיפת אלמנט חדש למחסנית - האלמנט נכנס לראש המחסנית.

2. רוטינה היא קטע קוד שמבצע פעולה מסוימת וניתן לקרוא לו ממקומות שונים בתכנית. היא מאפשרת חלוקה של הקוד לקטעים קטנים נוחים לתחזוקה ושימוש חוזר. הרוטינה מאפשרת לנו לשמור על קוד יעיל ולא מסורבל, וחוסכת תאים בזיכרון. הרוטינה תלויה במחסנית - כאשר קוראים לפונקציה, הכתובת ממנה ה – PC צריך להמשיך לאחר הקריאה לפונקציה וסיומה נשמרת במחסנית, וככה בסיום הפונקציה התוכנית ממשיכה להריץ את הקוד מהשורה הנכונה.

3. פונקציית מאקרו היא פונקציה שמוגדרת בקוד, בזמן הפיתוח, על ידי שמה. במהלך הקומפילציה, הפונקציה נכתבת כולה לתוך התוכנה. כלומר, כאשר יושב המהנדס ומבצע את התכנות, הוא ירשום רק את שם הפונקציה בקוד שלו, ואילו בתהליך הקומפילציה, הקומפיילר ירשום כל פעם שרשום את שם הפונקציה שוב ושוב את כל הפקודות שמרכיבות אותה. היתרון המרכזי בה על פני רוטינה היא שהשימוש בה חוסך זמן והחיסרון המרכזי הוא שהיא צורכת יותר זיכרון מרוטינה.

טבלת יתרונות וחסרונות בין פונקציית MACRO לבין רוטינה:

קריטריון	רוטינה	Macro
זמן ביצוע	קריאה לרוטינה דורש קריאה לכתובת, הכנת משתנים, שמירת מצב וחזרה- גורם לבזבוז זמן.	מתבצע כחלק מהתכנית, אין בזבוז של זמן בקריאה ובהפעלה של הפונקציה
זיכרון	חוסך מקום בזיכרון, קטע קוד מסוים נכתב פעם אחת עם כתובת ולא המון פעמים.	בזבזן בזיכרון, כל פעם שקוראים לפונקציית המקרו במהלך הקומפילציה, כל השורות נכתבות שוב ושוב.
תחזוק	קל לדבג מכיוון שהפונקציה היא קטע קוד נפרד.	יותר קשה לדבג מכיוון שהפונקציה היא חלק מקטע הקוד הכללי.
רקורסיה	תומך ברקורסיה, כולל שימוש במחסנית לניהול ושמירת הפרמטרים.	עלול להוביל לשגיאת קומפילציה בגלל לולאה אין סופית.

קבועים	פחות מתאים מכיוון שעלול לקחת הרבה זמן - גם הפעלה וקריאה לפונקציה, וגם הבאת הקבועים.	מתאים בגלל הזמן הקצר של הפעלת הפונקציה.

4. SCOPE של משתנה הוא הטווח שבו ניתן לגשת למשתנה בקטע קוד. הוא מגדיר היכן ניתן להשתמש במשתנה בתוך הפרויקט. אם נגדיר משתנה בתוך פונקציה לדוגמא, לא נוכל לקרוא לו בתוכנית הראשית.

5. נפרט בקשר לכל אחד מהמושגים בשאלה:

- משתנה גלובלי – משתנה שמוגדר בRAM או בFLASH בתוכנית הראשית, ונגיש לכל אורך ריצת התכנית לכל קטע קוד שמזמן אותו. לדוגמה, כתובת המשמשת לשמירת מקום ב – RAM עבור התוצאה (שומרת באמצעות DS16 1 לדוגמה).

- משתנה לוקאלי - משתנה המוגדר רק לאותה פעולה או קטע קוד ספציפי. לא ניתן לקרוא לו מכל נקודה בקוד. המשתנה הלוקאלי נוצר עבור קטע הקוד הספציפי או הפעולה, ונמחק בסופה. לדוגמה: שימוש ב – LOCAL כאשר יוצרים משתנה לוקאלי בתוך פונקציית מאקרו.

- קבוע בזיכרון - קבוע הוא ערך שנשמר בזיכרון הROM (Read Only Memory) – זיכרון שלא ניתן לשנותו וערכו הוא קבוע.

- קבוע מסוג text - קבוע אשר אינו מאוחסן בזיכרון והוא חלק מהתוכנית עצמה. התוכנה אינה צריכה ללכת להביא אותו מתא מסוים אלא הוא כתוב כבר כחלק מהפקודה. לדוגמה:mov #5, R4 : בפקודה זו, קבוע הטקסט הוא 5, ובתהליך הקומפילציה המחשב יהפוך את הקבוע הזה ל101, והוא יהיה שם כחלק מהתוכנית. לא יהיה צורך לגשת ולהביא אותו מתא בזיכרון, ולכן השימוש בקבוע זה הוא מהיר יחסית.

- רגיסטר- רגיסטר הוא יחידת עיבוד שנמצאת בCPU, לכן הגישה אליו מהירה מאוד, ומשמשת לאחסון זמני של נתונים לצורך עיבודם. לדוגמה: אחסון אורך של מערך נתון בתוך רגיסטר, וכך הרגיסטר משמש כמונה איטרציות.

נתונים עבור הקוד שכתבתנו:
זמן ריצה: התוכנית מבצעת 150 מחזורי שעון (לפני ביצוע הלולאה האינסופית הראשונה). 150 כפול 0.954 מיקרו שניות שווה ל – **143.1 מיקרו שניות**.

גודל התוכנית: התוכנית מתחילה בכתובת 0x3100 (תחילת ה – FLASH) ומסתיימת בכתובת 0x3142.
גודלה באקסדצימלי הוא 42 בתים, שבעשרוני מדובר ב – **66 בתים**.