משרד החינוך
האגף לתכנון ולפיתוח
תכניות לימודים

המרכז לטכנולוגיה
חינוכית (מט"ח)

המרכז הישראלי לחינוך
מדעי-טכנולוגי
ע"ש עמוס דה-שליט

האוניברסיטה הפתוחה
בית הספר לטכנולוגיה

מבוא למערכות מחשב ואסמבלי

מדריך למורה

אישור משרד החינוך
אישור מס' נ/4127

כתיבה
שרה פולק

קריאה וייעוץ מדעי-פדגוגי
ד״ר צבי פירסט

עריכה לשונית
יעל מילר

הפקה
אביבה אבידן

עיצוב עטיפה
אבי חתם

משרד החינוך
האגף לתכנון ולפיתוח
תכניות לימודים

המרכז הישראלי
לחינוך מדעי-טכנולוגי
ע״ש עמוס דה-שליט

המרכז לטכנולוגיה
חינוכית (מטח)

האוניברסיטה הפתוחה
בית-הספר לטכנולוגיה

מקט 1043313

מהדורת ניסוי

© מהדורת תשס״ז – 2006. כל הזכויות שמורות למשרד החינוך.
בית ההוצאה לאור של המרכז לטכנולוגיה חינוכית, קרית משה רואו, רח׳ קלאוזנר 16, רמת-אביב, ת״ד 39513, תל-אביב 61394.
The Centre For Educational Technology, 16 Klausner St., Ramat-Aviv, P.O.Box 39513, Tel-Aviv, 61394. Printed in Israel.

זכויות הקניין הרוחני, לרבות זכויות היוצרים והזכות המוסרית של היוצר/ים בספר זה מוגנות. אין לשכפל, להעתיק, לסכם, לצלם, להקליט, לתרגם, לאחסן במאגר מידע, לשדר או לקלוט בכל דרך או בכל אמצעי אלקטרוני, אופטי, מכני או אחר, כל חלק שהוא מספר זה. כמו כן, אין לעשות שימוש מסחרי כלשהו בספר זה, בכולו או בחלקים ממנו, אלא אך ורק לאחר קבלת רשות מפורשת בכתב ממטח (המרכז לטכנולוגיה חינוכית).

תוכן העניינים

פרק 1 –	המחשב הסִפרתי (הדיגיטלי)	5
פרק 2 –	ייצוג ידע במחשב	19
פרק 3 –	פעולות חשבון על ייצוג בינארי במחשב	23
פרק 4 –	שפת אסמבלי והמודל התכנותי של מעבד 8086	27
פרק 5 –	תכנות בסיסי בשפת אסמבלי	31
פרק 6 –	שיטות מיעון, מערכים ורשומות	45
פרק 7 –	מחסנית, פרוצדורות ומקרו	53
פרק 8 –	קלט פלט ופסיקות	59
פרק 9 –	מחרוזות	77
פרק 10 –	ארכיטקטורה של מעבדים מתקדמים	81

פרק 1
המחשב הסִפרתי (דיגיטלי)

משך הלימוד

שלוש שעות לימוד עיוני

מטרות

1. הכרת התלות ההדדית בין חומרה לתוכנה
2. הכרת המודל של מחשב מופשט ושל מבנה ההוראות בשפת מכונה
3. הכרת מחזור הבאה-ביצוע של ביצוע הוראה

נושאים עיקריים

1. חומרה ותוכנה והקשר ביניהן
2. מודל המחשב המופשט וביצוע הוראות

מושגים עיקריים

- מחשב, חומרה ותוכנה
- שפת מכונה, שפת אסמבלי, שפות נמוכות
- שפות עיליות
- תרגום תכנית לשפת מכונה: אסמבלר, מהדר, מפרש
- ארכיטקטורה של מחשב
- מודל מחשב המבוסס על ארכיטקטורת פון-ניומן
- מבנה המעבד והזיכרון
- מבנה הוראה בשפת מכונה
- מחזור הבאה-ביצוע
- תקשורת בין יחידות המחשב – הפסים והשעון

הידע שירכשו התלמידים בפרק זה יכשיר אותם:

1. להסביר את המושגים: חומרה, תוכנה, שפות נמוכות, שפות עיליות, תרגום תכנית לשפת מכונה וארכיטקטורה של מחשב.
2. לתאר באופן מפורט מחזור הבאה-ביצוע של הוראה ולהציג את מצב הזיכרון ומצב האוגרים בכל שלב.
3. לתכנן סוג חדש של הוראת מכונה במחשב הפשוט.
4. לתאר מבנה מחשב בסיסי ואת האופן שבו מתבצעת התקשורת בין היחידות השונות שלו.

פעילויות לתלמיד

פעילות 1 – מבוא למבנה המחשב ולשפת מכונה

הרחבה

א. שימוש במדמה (סימולטור)

יש תוכנות חזותיות המדמות מודל פשוט של מחשב כמו מודל "Little Man Computer", שתוכננו כדי לעזור לתלמידים להבין מהו מחזור הבאה-וביצוע באמצעות החזיית (ויזואליזציה) מרכיבי מחשב בסיסיים (מעבד, זיכרון ועוד). ההדמיה כוללת שפת מכונה בסיסית, שבאמצעותה אפשר לכתוב תכניות ולהתבונן בתהליך הרצה של תכנית במחשב. יש גרסאות רבות למימוש המודל הזה. להלן קישור לשתי גרסאות כאלה, שניתן להורידן ללא תשלום, המכילות תיעוד והסברים על אופן ההפעלה:

1. http://www.herts.ac.uk/ltdu/projects/mm5/
   תוכנה להורדה ולהרצה במחשב

2. http://www.cba.uri.edu/faculty/vvm/index.html
   תוכנה להרצה מהרשת (בתנאי שקיימת סביבת הרצה של JAVA)

ב. התפתחותם של המחשבים

הנושא הזה יכול להיות נושא ללמידה עצמית. להלן סיכום על התפתחותם של מחשבים וכן מקורות נוספים למידע בנושא.

ב.1 באינטרנט ישנם מקורות רבים בעברית ובאנגלית המכילים רקע על התפתחות המחשבים וכן מידע על אישים מפורסמים בתחום המחשוב.

להלן דוגמאות:

http://www2.eitan.ac.il/pioneers/fram.htm

http://www2.eitan.ac.il/comphistory/

"מנהרת הזמן" של חברת אינטל מציגה את ההתפתחות של מעבדי אינטל

http://www.intel.com/il/intel/timeline/timeline.htm (בעברית)

http://www.intel.com/museum/online/hist_micro/hof/index.htm?iid=intelmuseum+home_behof& (באנגלית)

מקור נוסף באנגלית:

http://www.kids-online.net/learn/clickjr/clickjr.html

אתר זה מציג את המבנה של המחשב ומתאר את החומרה. באתר מוצגת תמונה של המחשב ללא המכסה. הקלקה על רכיב כלשהו מעלה כיתוב וכן את התמונה של הרכיב.

ב.2 משימת חקר-רשת על התפתחות המחשבים.

במשימת חקר-רשת (WebQuest) התלמידים מתבקשים לבנות מצגת המתארת את הרגעים החשובים בהיסטוריה של התפתחות המחשבים. להלן דוגמה של אתר באנגלית, המציג משימת חקר-רשת ומספק הפניות למקורות אחרים ברשת. באתר הנחיות למבנה העבודה ודרכים להערכתה.

http://www.berksiu.k12.pa.us/webquest/mccaffrey/default.htm

התפתחות המחשבים (תקציר)

תקציר זה מתאר את התפתחות המחשבים ושם דגש על הטכנולוגיה, הארכיטקטורה והתוכנה.

המונח ***מחשב*** שימש בעבר כינוי לאנשים שביצעו חישובים שונים, כגון: הפקת טבלאות ניווט לספינות, חישוב זמני גאות ושפל, קביעת מיקום כוכבים ועוד. עבודתם הייתה משעממת ושגרתית ופעמים רבות הם שגו בחישוביהם. לכן רבו הממציאים שחיפשו דרך לבנות מכונות חישוב מהירות ומדויקות.

מכונת החישוב הראשונה הומצאה בשנת 1642 על-ידי בלז פסקל (Blaise Pascal) והיא ביצעה פעולות חיבור וחיסור בלבד. פסקל (שעל שמו נקראת שפת התכנות פסקל) ביקש לעזור לאביו פקיד המס, לבצע את חישובי המס בצורה מהירה. בגיל 19 הוא בנה מחשבון שנקרא בשם Pascaline שעיבד מספרים עשרוניים. המחשבון היה מורכב מסדרה של גלגלי שיניים, בדומה למנגנון של שעון מכני. ברגע שגלגל אחד (שייצג מאות) הסתובב סיבוב מלא, הוא הפעיל את הגלגל הסמוך לו (שייצג עשרות) ואשר הסתובב רק עשירית צעד. בעקבות הפיתוח המוצלח של המחשבון נבנו עוד 50 מכונות דומות.

לאחד כשלושים שנה (בשנת 1673), גוטפריד וילהלם לייבניץ (Gottfried Wilhelm Leibnitz) בנה מחשבון משוכלל יותר, שביצע את ארבע פעולות החשבון הבסיסיות (חיבור, חיסור, כפל וחילוק). המחשבון של Leibnitz לא היה בנוי מגלגלי שיניים אלא מתוף, שבהיקפו נחרצו עשרה חריצים בסדר עולה (כמו מדרגות – משום כך הוא נקרא Step reckoner). באמצעות התוף הזה בוצעו חישובים של מספרים עשרוניים.

המחשב הראשון שניתן לתכנת בו תכניות למטרות כלליות, תוכנן על-ידי מתמטיקאי אנגלי בשם צ׳רלס בּאבּג׳ (Charles Babbage) במאה ה-19. באותן שנים (1822) פרסמה הממשלה הבריטית שבעה ספרים שהכילו טבלאות ניווט לשימושם של יורדי ים, אולם זמן קצר לאחר מכן פורסמה רשימה של כ-1000 תיקונים!

בבאבג' הציע לבנות מכונת חישוב שנקראה "מכונת הֶפְרשים" (Difference Engine). ממשלת בריטניה הסכימה לממן את הפרויקט כי היא קיוותה שהמכונה הזו תוכל להפיק בצורה אוטומטית טבלאות ניווט מדויקות ואמינות יותר. התכנון והבנייה של המכונה התארכו, ולאחר עשר שנים, כשהפרויקט היה עדיין רחוק מסיום, הפסיקה ממשלת בריטניה את המימון והעבודה על הפרויקט נפסקה. אולם בבאבג' לא נרתע, ופנה לתכנן את המכונה הבאה.

בשנת 1842 החל בבאבג' לתכנן את "המכונה האנליטית" (Analytic Engine), שנועדה לתכנות פתרונות של בעיות כלליות. גם מכונה זו לא נבנתה מעולם אבל התכנון המפורט שערך בבאבג' משמש עד היום כמסגרת לארכיטקטורה ולתכנון של המחשבים המודרניים.

עיקרון חשוב שהניח בבאבג' בתכנון המכונה האנליטית היה ההפרדה בין הנתונים שיש לעבד לבין הפעולות שיש לבצע עליהם. עיקרון זה אִפשר גמישות בביצוע הפעולות או במילים אחרות, אִפשר לכתוב תכניות שונות לעיבוד הנתונים.

המכונה של בבאבג' הורכבה מארבע יחידות עיקריות:

- **ביחידות הקלט והפלט** השתמש בבאבג' בכרטיס מנוקב שפותח בשנת 1801 על-ידי ז'וזף-מרי ז'אקרד (Joseph-Marie Jacquard) לשימוש בנולים לאריגת בדים. הכרטיס היה לוח עץ מנוקב בחורים ודרך חורים אלה הועברו חוטים שיצרו את דוגמת האריגה המתוכננת. בבאבג' השתמש ברעיון הזה כדי להציג כל הוראת חישוב בכרטיס מנוקב אחד, שבו תבנית החורים הגדירה את הפעולה. באופן דומה הוא השתמש בכרטיסים גם לקליטת נתונים ולהצגת הפלט.
- **יחידת האחסון** (Store) היא יחידת הזיכרון שבה נשמרו ההוראות והנתונים. בבאבג' השתמש בצירים ובגלגלי שיניים לבניית יחידה זו. ביחידה היה אפשר לאחסן עד 100 מספרים בני 40 ספרות כל אחד.
- **יחידת העיבוד** (Mill) היא "לב" המחשב. בה "נארגו" או עובדו הנתונים בהתאם להוראות. יחידת העיבוד הורכבה ממאות צירים אנכיים ומאלפי גלגלי שיניים וגובהה היה כשלושה מטרים.

מבנה המכונה של בּאבּג' אִפשר לא רק לבצע בה פעולות חשבון באופן סדרתי (בדומה למחשבונים) אלא גם לשנות את סדר הביצוע של ההוראות – דילוג על כמה הוראות קדימה או חזרה של כמה צעדים אחורנית. למעשה, במכונה זו מומשו הפעולות if-then-else ובוצעה לולאת while-do. בהעדר אלו לא היה אפשר לשנות את אופן הביצוע של התכנית בזמן ריצת המחשב.

בּאבּג' אמנם תכנן את הארכיטקטורה ואת החומרה של המחשב, אבל את התוכנה להפעלתו פיתחה עדה ביירון (Ada Byron) המוכרת גם בשם הרוזנת מלאבלייס, שהייתה בתו של המשורר האנגלי לורד ביירון. ביירון הייתה בת 19 כששמעה על המכונה שבּאבּג' מתכנן. בהיותה מתמטיקאית מחוננת היא הוקסמה מהרעיון ונפגשה עם בּאבּג'. בּאבּג' הציג בפניה את עקרונות המכונה האנליטית. לאחר שלמדה את מבנה המכונה שעדיין לא נבנתה, החלה ביירון לכתוב תכניות המציעות פתרון של בעיות שונות באמצעות המכונה של בּאבּג'. בזכות תרומתה זו קיבלה ביירון את התואר **המתכנתת הראשונה בהיסטוריה** ועל שמה נקראה שפת התכנות **"עדה"**. רבים טוענים כי באבֶּג' נכשל כי הטכנולוגיה של זמנו לא אִפשרה בנייה של מכונה אנליטית.

פריצת הדרך הבאה הייתה בארה"ב. בשנת 1790 החלו לערוך בארה"ב מפקד תושבים כל עשר שנים כדי לקבוע כמה קולות על כל נציג לקבל כדי להיכנס לקונגרס. במפקדים הראשונים היה מספר התושבים קטן יחסית וביצוע המפקד וחישוב התוצאות ארך חודשים ספורים. במשך הזמן גדלה אוכלוסיית ארה"ב מאוד ובשנת 1880 נמשך חישוב תוצאות המפקד שבע שנים! הממשל בארה"ב חשש שביצוע המפקד שיבוא אחריו יארך יותר מעשר שנים ולכן פרסם מכרז שבו הציע פרס לממציא שיטה יעילה יותר לעריכת מפקד התושבים של 1890. בפרס זה זכה הרמן הולריית' (Herman Hollerith). הוא הציע מכונה שהייתה מבוססת על הרעיון של ז'אקרד – שימוש בכרטיסים מנוקבים להזנת נתונים למכונה.

המכונה נקראה בשם **Hollerith desk**. היא הייתה בנויה מקורא כרטיסים מנוקבים וממערכת גלגלי שיניים שספרה את מספר החורים שבכרטיסים. היו בה מחווני שעונים (בדומה למד מהירות של מכונית) שנקבעו כדי להציג את תוצאות הספירה של הקולות. למעשה, היה זה המחשב הראשון שנועד לשימוש מסחרי מסוים. באמצעות המכונה שבנה הולריית', הסתיים חישוב מפקד האוכלוסין לאחר שלוש שנים בלבד.

בהמשך הקים הולריית' חברה בשם International Business Machines הידועה היום בשם **IBM**. למעשה, כרטיסים מנוקבים היו בשימוש כאמצעי הקלט להזנת נתונים ותכניות עד תחילת שנות השבעים של המאה העשרים.

היום מתנהלים ויכוחים רבים בקשר לשאלה מיהו המחשב למטרות כלליות שנבנה ופעל ראשון. למעשה, בשנת 1937 מדען גרמני בשם קונרד זוס (Konrad Zuse) תכנן ובנה את המחשב המכני הראשון שפעל. זוס השתמש לראשונה במכונת עיבוד מספרים שהיו מורכבים משני סימנים: אפס ואחד (0, 1) בלבד, שמומשו באמצעות רכיבים אלקטרו-מכניים שנקראו ממסרים. זוס יישם פעולות חישוב המבוססות על ייצוג זה. חשבון בשני סימנים נקרא ***חשבון בינארי*** ובנושא זה נעסוק בפרקים הבאים. פרסום המכונה של זוס התעכב בזמן מלחמת העולם השנייה ורק בשנות השישים של המאה העשרים המכונה נודעה ברבים. לכן המחשב שזכה לתואר המחשב המכני הראשון שפעל על-פי העקרונות של באבג', היה המחשב שנקרא **Harvard Mark** I. מחשב זה תוכנן בשנת 1943 על-ידי הווארד אייקן (Howard Aiken) בשיתוף חברת IBM.

למחשבים המכניים היו כמה חסרונות בולטים: מהירות העיבוד שלהם הוגבלה בגלל שימוש ברכיבים מכניים (כמו גלגלי שיניים). רכיבים אלה היו מסורבלים, לא אמינים ויקרים מאוד לבנייה. בתחילת המאה העשרים התחילו להשתמש בטכנולוגיה המבוססת על רכיבים אלקטרוניים וכך הסתיימה התקופה של שימוש ברכיבים המכניים.

בעיקרון, אפשר להבחין בשישה דורות של מחשבים אלקטרוניים. כל דור התאפיין בהתפתחות דרמטית בשלושה תחומים:

א. בטכנולוגיה שבה השתמשו לבניית המחשב.

ב בארכיטקטורה של המחשב.

ג. בתוכנה.

הדור הראשון של מחשבים אלקטרוניים (1937 – 1953)

הטכנולוגיה שבה השתמשו לבניית המעגלים האלקטרוניים בדור הראשון הייתה מבוססת על שפופרות ריק (Vacuum tube) ששימשו כמתגים הנמצאים באחד משני מצבים: מצב דלוק או כבוי. (בדומה לכרטיס המנוקב שבו שני המצבים נקבעו על-ידי הימצאות חור במקום מסוים או חסרונו שם).

המעבר של שפופרות ריק ממצב אחד למצב השני הייתה פי אלף יותר מהירה ממהירות המעבר של מתגים מכניים כמו גלגלי שיניים. גם פעולת השפופרות הייתה אמינה יותר משום שהן לא נטו להישחק ולהיתקע כמו מתגים מכניים.

בחלק ניכר של תקופה זו התחוללה מלחמת העולם השנייה והממשלות השונות התעניינו בפיתוח מכונות שיסייעו למאמץ המלחמתי. בבריטניה עסק החוקר אלן טיורינג (Alan Turing) בפיתוח מחשב שכונה בשם **Colossus** אשר שימש לפיצוח תשדורות מוצפנות של הצבא הגרמני. טיורניג ידוע בעיקר בשל פיתוח של מודל מופשט של אופן פעולת המחשב הנקרא ***מכונת טיורינג*** (שנת 1936). באותה תקופה בגרמניה המשיך זוס לפתח דגמים נוספים של המחשב Z1 ובשנת 1941 הוא הצליח לבנות את המחשב שכונה Z3, שהיה מחשב אלקטרוני.

במקביל, צבא ארה״ב התעניין במכונות שיסייעו לחישוב מסלולי ירי של תותחים שהיו מוצבים בספינות מלחמה. הכורח לשלוח ספינות מלחמה למקומות שונים יצר צורך דחוף בטבלאות לחישוב מסלולי ירי של תותחים שהיו מוצבים בספינות. המחשב נבנה והורכב מרכיבים אלקטרוניים והוא כונה בשם **ENIAC** (Electronic Numerical Integrator and Computer). מחשב זה תוכנן ונבנה במימון צבא ארה״ב, עלידי שני פרופסורים מאוניברסיטת פנסילבניה: ג׳ון מוסי (John Mouchy) ו-פרספר אקרט (Presper Eckert), לאחר שהבטיחו כי יוכלו לבנות מחשב שיהיה מסוגל להחליף את כל ה״מחשבים״ שהיו עד אז בשימוש הצבא לחישוב טבלאות של מסלולי ירי.

בניית המחשב החלה ב-1943 והסתיימה לאחר המלחמה (ב-1945) ולכן הוא לא נכנס לשימוש לחישוב של מסלולי ירי, כמתוכנן. לעומת זאת השתמשו בו לפיתוח פצצת המימן. לאחר מכן שימש מחשב ה-**ENIAC** במחקרים לחיזוי מזג אוויר, ליצירת מספרים אקראיים ועוד. המחשב שנבנה מילא חדר בגודל של 12x6 מטר, הוא שקל 30 טון והכיל כ-18,000 שפופרות ריק שהפיצו חום רב ולכן נדרשה בחדר מערכת קירור מיוחדת.

הארכיטקטורה שלו הייתה מבוססת על הארכיטקטורה של המחשב שתכנן בּאבּג' ואשר כללה יחידת קלט, יחידת פלט, יחידת זיכרון ויחידת עיבוד. העיבוד נעשה באמצעות מספרים עשרוניים. כאשר הוצגה המכונה בפני האנשים שעסקו בצבא ארה״ב בתפקיד של ״מחשבים״ אמרה אחת מהם: ״אני המומה מכך שכל הציוד הזה דרוש כדי להכפיל 5 ב-1000״. כתיבת התכנית התבצעה במחשב זה בחומרה. כלומר, ההוראות נקבעו על-ידי מתגים וחוטים שיצרו את החיווט של המעגלים החשמליים הדרושים לביצוע הפעולה.

לאחר שפיתוח ה-**ENIAC** הסתיים החלו המפתחים לחפש דרכים חדשות כדי לשפר את תהליך התכנות של המחשב שארך לעיתים ימים שלמים. לאחר הצטרפות מתמטיקאי בשם ג׳ון פון-ניומן (John von Neumann) לצוות הפיתוח נבנה ה-**EDVAC** (Electronic Discrete Variable Automatic Computer). המחשב הזה החל לפעול בשנת 1951 והארכיטקטורה שלו התבססה על גישה חדשה. הרעיון המהפכני שהגו פון-ניומן וצוותו הוא שתכנית יכולה להיות מאוחסנת כפי שמאחסנים נתונים. עד אז תכנית פעלה כיחידת עיבוד שצריכה לקרוא שתי קבוצות נפרדות של כרטיסים מנוקבים – האחת להוראות והשנייה לנתונים. ההמצאה החדשה אפשרה למחשב לעדכן בעצמו את קבוצת ההוראות ולטפל בהן כמו בנתונים. כך הצליח המחשב גם לכתוב תכניות שיַעבְּדו תכניות אחרות (למשל מהדרים). ממציאי המחשב הזה החליטו לראשונה להשתמש בייצוג נתונים בשיטה הבינארית (שימוש בשני סמלים בלבד: 0 ו-1). כיום רוב המחשבים משתמשים בארכיטקטורה זו הנקראת **ארכיטקטורה בשיטת פון-ניומן** (Von Neumann Architecture).

הרעיון של תכנית מאוחסנת (stored program) היווה נקודת מפנה גם בתחום התוכנה. מאותו זמן התאפשר לכתוב תכניות בשפת מכונה. כלומר, לרשום את ההוראות במספרים לפי קודים של פעולות המחשב ולשמרם בזיכרון.

מאוחר יותר, בתחילת שנות החמישים של המאה העשרים, החלו לכתוב בשפה סימבולית שכבר הצגנו בפניכם – **שפת אסמבלי**. תחילה כתבו את ההוראות בשפת אסמבלי בקודים מנמוניים ולאחר שכתבו את התכנית המירו אותה באופן ידני לשפת מכונה. מאוחר יותר נכתבה תכנית בשם אסמבלר שתפקידה היה לבצע את ההמרה משפת אסמבלי לשפת מכונה. באותו דור המחשבים היו גדולים מאוד ומחירם היה גבוה ולכן יצרו רק מכונה אחת מכל גרסה. מחשבים היו בעיקר בשימוש של ממשלות, אוניברסיטאות או של חברות גדולות מאוד.

הדור השני של מחשבים אלקטרוניים (1954-1962)

בשנת 1948 חלה התפתחות טכנולוגית דרמטית – המצאת הטרנזיסטור. רכיב זה החליף את השימוש בשפופרות הריק (משנת 1954 ואילך). השימוש בטרנזיסטורים במחשב הגדיל את מהירות העיבוד ל-6000 עד 3 מיליון פעולות בשנייה. גודלו של המחשב הוקטן לארון שמידותיו היו כ-2×1.2 מטר ומשקלו צומצם לכמה מאות קילוגרם בלבד.

גם הטכנולוגיה שבה השתמשו לבניית זיכרונות הוחלפה ואז החל שימוש ברכיבים אלקטרוניים שאפשרו גישה אקראית לכל נתון בזיכרון. בגישה אקראית זו אפשר לגשת ישירות לנתון מסוים (לפי כתובת, בדומה למציאת שיר בתקליטור). לפני כן השתמשו באמצעים שאפשרו גישה סדרתית בלבד בדומה למציאת שיר ברשמקול. כלומר, כדי לגשת לכתובת מסוימת בזיכרון היה צריך לעבור על כל הנתונים שבכתובות שקדמו לכתובת זו.

התפתחות חשובה חלה גם בארכיטקטורה של המחשבים. נוספו רכיבים שאפשרו גישה מהירה לאיברי מערך. בארכיטקטורה הקודמת התאפשר לגשת לאיברים בזיכרון רק על-פי רישום של כתובת מפורשת. כדי לגשת לאיבר הבא במערך היו צריכים לכתוב קוד שיעדכן את עצמו בזמן הריצה (טכניקה שהקשתה מאוד על איתור תקלות). בארכיטקטורה החדשה היה אפשר לעבד את הכתובת ולחשב את הכתובת של האיבר הבא במערך (i=i+1).

כמו כן החל שימוש ביחידות חומרה לביצוע חישובים על מספרים ממשיים. לפני-כן בוצעו חישובים אלה על-ידי תוכנה ותהליך החישוב היה איטי מאוד. באופן עקרוני ביצוע פעולה בחומרה מהיר יותר מפני שלא נדרש זמן תרגום לשפת מכונה ולא נדרש זמן לביצוע מחזורי הבאה ולביצוע של ההוראות.

במקביל, פותחו בתחום התוכנה השפות העיליות הראשונות. פורטרן **Fortran** הייתה השפה העילית הראשונה (פותחה בשנת 1958). היא שימשה לבניית יישומים מדעיים. בשנת 1959 פותחה שפת קובול **Cobol** אשר שימשה ליישומים מסחריים. המעבר משפת אסמבלי לשפה עילית אִפשר רמה נוספת של הפשטה דבר שהֵקֵל על המתכנתים לכתוב תכניות שלא היו תלויות במבנה של מחשב מסוים, ולכן היה אפשר להריצן על מחשבים שונים.

הדור השלישי של מחשבים אלקטרוניים (1963-1972)

המגמה של מזעור המחשב נמשכה וההתפתחות החשובה ביותר (בשנת 1961) הייתה פיתוח של מעגלים מוכללים (**IC** – Integrated Circuits). מעגל מוכלל הוא רכיב הבנוי מפיסה של מוליך למחצה שעליה הורכבו מאות טרנזיסטורים.

המזעור גרם להגדלת מהירות העיבוד להקטנת עלויות האנרגיה ובעקבות זאת להוזלת מחיר המחשבים באופן משמעותי. ההוזלה אפשרה גם לעסקים בינוניים לרכוש מחשבים לעיבודים עסקיים. שיפור נוסף חל בתחום רכיבי הזיכרון – האמצעים המגנטיים הוחלפו בזיכרונות העשויים ממוליכים למחצה. השימוש בטכנולוגיה הזו הגדיל את קיבולת הזיכרון ואת מהירות הגישה אל הנתונים.

בתחום הארכיטקטורה של המחשב פותחו טכניקות לתכנון מעבדים מורכבים. כך פותחו טכניקות של תכנות מקבילי (הרצת כמה משימות על מעבד יחיד). מתכנני המחשב התחילו להשתמש בטכניקה של הרצת תכניות שונות במקביל על-ידי שימוש ביחידות פונקציונליות ובחפיפה של פעולות IO ושל CPU, הן בזרם ההוראות והן בזרם הנתונים. כך למשל, אפשר היה לבצע בו זמנית שתי הוראות: הוראה אחת של קלט/פלט והוראה שנייה של חישוב. כל אחת מההוראות הללו התבצעה ברכיב אחר של המעבד.

בתחום התוכנה – הפיתוח המשמעותי היה בניית מערכת ההפעלה הראשונה **unix**. מערכת ההפעלה היא תוכנה האחראית על ניצול משאבי המחשב בצורה יעילה והוגנת. מערכת ההפעלה מקלה על עבודת המתכנת הכותב בשפה עילית. היא פוטרת אותו, למשל, מהצורך להגדיר כיצד לפנות ליחידות הקלט/פלט כדי לקרוא או לכתוב נתונים, או כיצד לפנות לזיכרון כדי לקרוא קובץ.

היום קשה לדמיין את המחשב ללא מערכת הפעלה. למעשה, זו התוכנה הראשונה שהותקנה במחשב ושתפקידה לנהל את החומרה. כתוצאה מכך, הפעולות שיש לבצע כדי לאתחל מחשב או כדי לקרוא קובץ מהזיכרון, שקופות למשתמש.

הדור הרביעי של מחשבים אלקטרוניים (1973-1984)

שיפור נוסף בטכנולוגיה חל בעקבות מזעור נוסף של המחשב ושימוש במעגלים מוכללים, שהכילו בדור המחשבים הזה אלף רכיבים על פיסה אחת (**VLSI** – Very Large Scale Integration). המזעור אִפשר לתכנן ולבנות מחשב שלם (מעבד, זיכרון ובקרי קלט/פלט) על פיסת סיליקון אחת (chip). כך הופיע בשנת 1981 המחשב האישי הראשון של חברת IBM. בנוסף, החלה בניית הזיכרון תוך שימוש בזיכרונות הבנויים ממוליכים למחצה, וכך הוגדלה כמות הנתונים שהתאפשר לאחסן והשתפרה מהירות הגישה לתא בזיכרון.

בתחום התוכנה הוחל בפיתוח שפות דקלרטיביות כמו תכנות פונקציונלי ותכנות לוגי ובעקבות כך פותחו יישומים רבים בתחום הבינה המלאכותית. בשפות דקלרטיביות המתכנת מגדיר בצורה מתמטית ולוגית מה צריך לחשב ואת פעולת פרטי החישוב הוא משאיר למהדר או למערכת ההרצה. לדוגמה, אפשר לנסח יחס המגדיר את סבא כאבא של אבא, ומערכת ההרצה תצטרך למצוא את הנתונים המקיימים יחס זה.

הדור החמישי של מחשבים אלקטרוניים (1985-1990)

טכנולוגיית הייצור של המעגלים המוכללים השתפרה בשנים אלה כך, שהתאפשר להרכיב מיליון רכיבים על פיסת סיליקון אחת (chip). שיפור זה שימש גם לייצור זיכרונות שהיו בנויים ממוליכים למחצה אשר הפכו להיות הרכיבים התקניים בכל מחשב. כך גדלו הן קיבולת הזיכרון והן מהירות הגישה לזיכרון.

בארכיטקטורה של המחשבים חלה התפתחות משמעותית שכללה שיתוף של מספר מעבדים שהריצו תכנית אחת. תחום זה נקרא **עיבוד מבוזר**. כך פיצחו את הגנום האנושי תוך שימוש במחשבים רבים. התפתחות נוספת חלה בפיתוח רשתות מקומיות של תקשורת מחשבים אשר בשלב זה שרתו בעיקר ארגונים וחברות עִסקיות.

בתחום התוכנה החל שימוש בשפות תכנות מונחות-עצמים כמו **smalltalk** ושפת **C++** שאפשרו להתמודד עם פיתוח יישומים גדולים (המורכבים ממאות אלפי שורות קוד). בתכנות מונחה-עצמים המבנה הבסיסי הוא עצם, שיש לו תכונות והתנהגות. כל תוכנה היא, בעצם, אוסף היררכי של אובייקטים רבים, המתקשרים אחד עם השני למערכת גדולה ומתואמת.

הדור השישי של מחשבים אלקטרוניים (מ-1991 ואילך)

המזעור של מעגלים מוכללים, הגידול בקיבולת הזיכרון ומהירות העיבוד אפשרו פיתוח של יישומים עתירי זיכרון כמו יישומי מולטימדיה (תמונות, קול ואנימציות). אבל ההתפתחות המשמעותית היא השימוש ברשתות תקשורת וברשת האינטרנט.

בתחום התוכנה פותחו שפות עיליות שאפשרו כתיבת יישומים לרשתות תקשורת וביניהן שפת **java** שפותחה בשנת 1995 על-ידי חברת SUN, ושפת **C#** שהוצגה על-ידי חברת מיקרוסופט בשנת 2000.

פרק 2
ייצוג מידע במחשב

משך הלימוד

שלוש שעות לימוד עיוני

מטרות

1. הכרת הייצוג הבינארי
2. הבנת מספרים, טקסטים ותמונות בייצוג בינארי
3. הבנת המגבלות של הייצוג הבינארי

נושאים עיקריים

1. ייצוג בינארי ושימור מידע
2. ייצוג מספרים שלמים ללא סימן בשיטה העשרונית, הבינארית, ההקסהדצימלית והאוקטלית
3. עקרונות הייצוג של מספרים ממשיים. (הערה – במהלך הקורס לא יוצגו פעולות על מספרים בשיטת הנקודה הצפה)
4. ייצוג תווים בקוד אסקיי ובקוד Unicode
5. עקרונות הייצוג של תמונות וקול
6. יחידות אחסון ומגבלות האחסון בהן

הידע שירכשו התלמידים בפרק זה יכשיר אותם:

1. להבין מהו ייצוג משמר מידע.
2. להסביר מתי ייצוג מידע הוא משמר מידע.
3. להסביר את שיטת הייצוג של מספרים שלמים ללא סימן ושל תווים ללא סימן.
4. להבין באופן עקרוני את הייצוג של קול ותמונה במחשב ולדעת להסביר מדוע ייצוג זה אינו משמר מידע.

5. להבין באופן עקרוני את השיטה לייצוג מספרים ממשיים ולדעת להסביר מדוע ייצוג זה אינו משמר מידע.
6. להכיר גדלים של יחידות אחסון.
7. לחשב את תחום הערכים (בשיטה העשרונית, הבינארית והקסדצימלית) שאפשרי לאחסן ביחידת אחסון נתונה.

הרחבה

א. שימוש במחשבון לחישובים בשיטות ספירה שונות

מערכת ההפעלה חלונות כוללת מחשבון המבצע פעולות חשבון בבסיסים: 2, 8, 10 ו-16.
להפעלת המחשבון:

בחרו בתפריט התחל ← כל התכניות ← תכניות עזר (accessorie) ← מחשבון

כדי לקבוע את מצב העבודה של המחשבון למצב עבודה מדעי בחרו בתפריט ראשי של המחשבון: view וקבעו את מצב העבודה כמצב "Scientific".

כך מאפשר המחשבון ביצוע של פעולות חשבון בשיטות הספירה הבאות: עשרוני, בינארי, הקסדצימלי ואוקטלי.

ב. הפעלה באמצעות משחק

המרה משיטת ספירה עשרונית לשיטת ספירה בינארית ולהפך:

המשחק כולל חמישה כרטיסים שעליהם כתובים 16 מספרים בין 1 ל-31. הכרטיסים ממוספרים במספרים הבאים: 1, 2, 4, 8, 16.

התלמיד מתבקש לחשוב על מספר מ-1 עד 31 ולהצביע על הכרטיסים שבהם כתוב המספר. חישוב פשוט של סכום מספרי הכרטיסים שנבחרו מגלה את המספר שעליו חשב התלמיד.

לדוגמה, נתונים חמשת כרטיסים הבאים:

כרטיס 1

1	3	5	7
9	11	13	15
17	19	21	23
25	27	29	13

כרטיס 2

2	3	6	7
10	11	14	15
18	19	22	23
26	27	30	31

כרטיס 4

4	5	6	7
12	13	14	15
20	21	22	23
28	29	30	31

כרטיס 8

8	9	10	11
12	13	14	15
24	25	26	27
28	29	30	31

כרטיס 16

16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31

אם תלמיד הצביע על הכרטיסים: 1, 4, 8 הוא בחר במספר 13.

העיקרון שעליו מבוסס המשחק:

- כל כרטיס מציין מיקום של ספרה בינארית ומכיל את כל המספרים שבהם ספרה זו היא 1.
- הכרטיס הראשון מכיל את הייצוג העשרוני של כל המספרים הבינאריים שבהם הספרה הכי פחות משמעותית בהם היא 1.
- הכרטיס השני מכיל את הייצוג העשרוני של כל המספרים הבינאריים שבהם הספרה במיקום השני (מימין) היא 1.
- הכרטיס השלישי מכיל את הייצוג העשרוני של כל המספרים הבינאריים שבהם הספרה במיקום השלישי (מימין) היא 1.

וכך הלאה.

ג. אתרים על שיטות קידוד ותקנים (באנגלית)

http://www2.eitan.ac.il/fonts/ (בעברית)

http://qcpages.qc.edu/~nixon/links/asciiUnicode.html (באנגלית)

http://www.cs.tut.fi/~jkorpela/chars.html (באנגלית)

פרק 3
פעולות חשבון על ייצוג בינארי במחשב

משך הלימוד

שלוש שעות לימוד עיוני

מטרות

1. הכרת הדרך לייצוג מספרים עם סימן (שיטת המשלים לשתיים)
2. ביצוע פעולות חשבון על מספרים בינאריים שלמים ללא סימן ועם סימן
3. הבנת הקשר בין מגבלות הייצוג במחשב ותוצאות החישוב (גלישה)

נושאים עיקריים

א. חיבור וחיסור של מספרים בינאריים ללא סימן

ב. כפל וחילוק של מספרים בינאריים ללא סימן

ג. שיטות ייצוג מספרים בינאריים עם סימן

ד. חיבור וחיסור של מספרים בינאריים שלמים עם סימן

ה. תחום הייצוג של מספרים בינאריים שלמים עם סימן וללא סימן

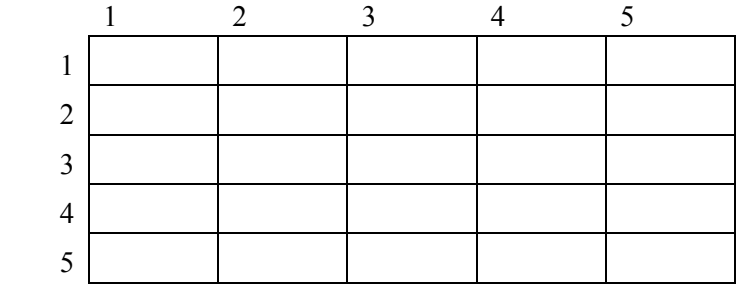
הידע שירכשו התלמידים בפרק זה יכשיר אותם:

1. להציג ולפענח מספר עם סימן בשיטת המשלים לשתיים.
2. לדעת לבצע פעולות חשבון בשיטה הבינארית, בעיקר חיבור וחיסור (המטרה העיקרית היא לדעת לקרוא תוצאות של עיבוד ולכן אין צורך להתעמק בביצוע פעולות חשבון אלא להתמקד בהצגת העקרונות).

הרחבה

א. הפעלה באמצעות תשבץ

ערכי התשבץ הבא נתונים בשיטה הבינארית. התלמידים מתבקשים למלא תשבץ זה בהתאם להגדרות הנתונות להלן:



מאוזן:

1. מספר ראשוני בין 11110 ל-100100
2. $\underline{111010}$ 10
3. $100\cdot(11+10)$
4. 101^{10}
5. $11\cdot10^{10}$

מאונך:

6. $110\cdot101$
7. $101-10^{101}$
8. $10\cdot11\cdot100+101$
9. $1+\underline{111100}$ 100

פתרון התשבץ:

	1	2	3	4	5
1	1	1	1	1	1
6	1	1	1	0	1
7	1	0	1	0	0
8	1	1	0	0	1
9	0	1	1	0	0

ב. נקודה צפה

אתר הכולל הסבר ותרגול עם פתרונות בנושא נקודה צפה (באנגלית):

http://www.nuvisionmiami.com/books/asm/3rd/workbook/floating_tut.htm

אתר הכולל יישום הממיר מספרים עשרוניים למספרים בשיטת נקודה צפה

http://babbage.cs.qc.edu/courses/cs341/IEEE-754.html

פרק 4
שפת אסמבלי והמודל התכנותי של מעבד 8086

משך הלימוד

שש שעות לימוד עיוני, ארבע שעות התנסות

מטרות

1. הכרת האוגרים
2. הכרת המאפיינים של הזיכרון : גודל, רוחב תא, כתובת, קריאה או קריאה/כתיבה
3. הכרת ארגון מרחב הזיכרון במעבד 8086
4. הבנת העקרונות של ארגון המקטעים בזיכרון ושל אופן חישוב הכתובות
5. הכרת מבנה ההוראה בשפת אסמבלי
6. הכרת מבנה התכנית בשפת אסמבלי והשימוש בהנחיות אסמבלר
7. הכרת טיפוסי משתנים והגדרתם

נושאים עיקריים

1 המודל התכנותי של מעבד 8086
2. ארגון הזיכרון במעבד 8086
3. מפת הזיכרון הראשי
4. חלוקת הזיכרון למקטעים (סגמנטציה)
5. כתיבת תכנית בשפת אסמבלי
6. הגדרת משתנים

מושגים עיקריים

1. עקרון התאימות
2. אוגרים

3. הזיכרון הראשי, הזיכרון המשני
4. מקטעים
5. כתובת מוחלטת, כתובת יחסית (היסט, אפקטיבית),כתובת בסיס
6. הוראה בשפת אסמבלי, הנחיית אסמבלר
7. טיפוס משתנה בשפת אסמבלי

הידע שירכשו התלמידים בפרק יכשיר אותם:

- לתאר את תפקיד האוגרים.
- להגדיר מאפיינים של זיכרון: כתובת, גודל זיכרון, גודל תא.
- לחשב גדלים של זיכרון.
- להמיר כתובת יחסית לכתובת מוחלטת.
- לכתוב שלד של תכנית פשוטה בשפת אסמבלי במודל small הכולל אתחול מקטע נתונים וסיומו והעברת בקרה למערכת ההפעלה
- לכתוב תכנית בשפת אסמבלי פשוטה המשתמשת בהוראות MOV ו-ADD.
- להגדיר משתנים בשפת אסמבלי.
- לחשב כתובת של משתנה.

פעילויות לתלמיד

- פעילות 2 – המודל התכנותי של מעבד 8086
- פעילות 3 – תהליך כתיבת תכנית בשפת סף והרצתה

הרחבה

א. אתר שבו מוצג המודל התכנותי של מעבדים: 8086, 80286, 80386, 80486, 80586 (באנגלית)

http://www.comsci.us/cpu/pm86.html

פתרונות של תרגילים נבחרים מפעילות 3

תרגיל 3

2048K = 2M

1T =

4096E =

תרגיל 4

בשיטת מיעון מקטעים אפשר:

א. להגדיר מקטע בגודל 64KB (כגודל הזיכרון במעבד 8085)

ב. לרשום כתובת ב-16 סיביות (כתובת במעבד 8085)

תרגיל 5

א. לא, כיוון שכדי לחשב כתובת פיזית עלינו לדעת את כתובת הבסיס של המקטע.

ב. להלן הכתובות הפיזיות של התא הראשון מחמשת המקטעים הראשונים:

40000
30000
20000
10000
00000

הכתובת היחסית של התא הראשון במקטע החמישי היא: 0000h

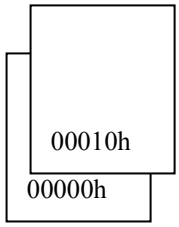
והכתובת הפיזית היא: 4000h

תרגיל 6

האיור הבא מציג את שני המקטעים.

הכתובת הרשומה בכל מקטע היא

כתובת הבסיס.



א. לא נכון, שני המקטעים חופפים.

ב. לא נכון, לא קיימת חפיפה מלאה.

ג. נכון, המקטעים חופפים. 64KB – 2B = 65534 שזה יותר מ-80%

תרגיל 9

שם משתנה	כתובת פיזית	כתובת יחסית
a	10000	0000
b	10001	0001
c	10003	0003

תרגיל 10

במקטע נתונים הוגדרו המשתנים הבאים:

```
.DATA
        x       DB
        y       DT
        q       DW
        t       DQ
```

א. גודל מקטע הנתונים הדרוש הוא:

1+10+2+4 = 17 byte

ב. 000Dh

ג. אי-אפשר משום שכתובת הבסיס של DS אינה ידועה.

פרק 5
תכנות בסיסי בשפת אסמבלי

משך הלימוד

12 שעות לימוד עיוני, שמונה שעות התנסות

מטרות

- הבנה של פתרון בעיות בשפת אסמבלי
- הבנת אופן המימוש של מבני תכנות המוכרים משפות עיליות בשפת אסמבלי
- הבנת אופן ביצוע של הוראות במחשב ושל ההשפעה של מגבלות המחשב על ביצוע חישובים

נושאים עיקריים

- מבנה הוראה בשפת אסמבלי (קוד פעולה, אופרנדים, טיפוסי אופרנדים מותרים, השפעה על הדגלים)
- שימוש בהוראות חיבור וחיסור: add, adc, sub, sbb, inc, dec, neg
- השפעת טיפוס אופרנד על תוצאת חישוב – גלישה
- השפעת החישוב במספרים שלמים ללא סימן ובמספרים שלמים עם סימן על אוגרי המצב (התייחסות מעמיקה לדגלים הבאים: SF, ZF, OF, CF)
- שימוש בהוראות קפיצה למימוש מבני בקרה if, case, while, repeat, for ושימוש בהוראה CMP ו-LOOP
- הוראות הקפיצה שבהן משתמשים הם:

jc, jnc, jz, jnz, js, jns, jo, jnoJC, JNC, JZ, JNZ, JS, JNS, JO, JNO

ja (jnbe), jb (jnae), jae (inb), jbe (jna), je, jne

jg (jnle), jl (jnge), jge (jnl), jle (jng), je, jne

- שימוש בהוראות כפל וחילוק:

mul, imul, div, idiv, cbw, cbd

- שימוש בהוראות לוגיות:

not, and, or, xor

- שימוש במיסוך באמצעות הוראות לוגיות לעיבוד נתונים
- שימוש בהוראות הזזה וסיבוב לעיבוד סיביות ושימוש בהוראות לפעולות כפל וחילוק

sal, sar, shl, shr, ror, rol, rcl, rcr

הידע שירכשו התלמידים בפרק זה יכשיר אותם:

- לכתוב תכנית בשפת אסמבלי הפותרת בעיה נתונה.
- לדעת איך להתאים טיפוסי משתנים לבעיה נתונה.
- להבין את מגבלות החישוב בהתאם לטיפוס המשתנים.
- לדעת לבחור הוראות בשפת אסמבלי המתאימות לבעיה נתונה. לדוגמה, בחירה של הוראת קפיצה מתאימה מתוך אוסף קיים.
- לממש מבני תכנות הכוללים תנאים, לולאות ולולאות כפולות.
- לקרוא ולעקוב אחרי ביצוע של תכנית בשפת אסמבלי.
- לבדוק תוצאות של חישובים חשבוניים ולחשב לוגיים על מספרים שלמים בבסיס בינארי ועשרוני.
- לאתר שגיאות בתכנית ולתקנן.
- להשלים הוראות חסרות בתכנית.

פעילויות לתלמיד

פעילות 4 – שימוש בהוראות העברה ובהוראות חשבוניות לחיבור ולחיסור

פעילות 5 – אוגר דגלים והוראות קפיצה

פעילות 6 – לולאות

פעילות 7 – הוראות לוגיות

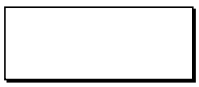
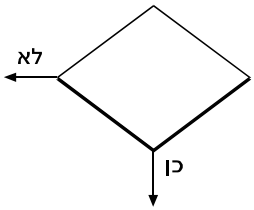
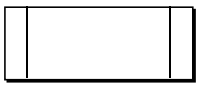
פעילות 8 – הוראות כפל וחילוק, הזזה וסיבוב

הרחבה

א. מעקב אחרי תכניות באמצעות תרשים זרימה

תרשים זרימה הוא תרשים גרפי המורכב מאוסף מוסכם של סימנים גרפיים. בעזרת הסימנים האלה אפשר לתאר את השלבים של תהליך התכנות, את הקשר בין השלבים האלה, את סדר השלבים ואת הפעולות המתבצעות בכל שלב ושלב. תיאור אופן הפעולה של המערכת בתרשים גרפי מאפשר להציג את התהליך בצורה תמציתית וברורה וקל לבצע בו שינויים או עריכה.

באיור 1 מתוארים הסמלים הנפוצים והשימושיים ביותר בתרשימי זרימה.

הסמל	המשמעות
	מציין התחלה או סיום של תרשים
	מציין פעולה של קלט או פלט נתונים
	מציין ביצוע הוראה ביצוע כגון השמה, חישוב וכדומה
	מצביע על הפעולה הבאה לביצוע
	מציין פעולת בקרה–בדיקת תנאי כלשהו שיכול להתקיים (תוצאה חיובית) או שיכול לא להתקיים (תוצאה שלילית)
	מציין קטע של תוכנית שאינו מפורט בשלב זה

איור 1
סמלים בתרשים זרימה

לדוגמה נתאר שימוש בתרשים זרימה לכתיבת תכנית שתפתור את הבעיה הבאה:

במפעל לייצור פלסטיק קיימת מערכת אלקטרונית שתפקידה לבקר את פעולת התנורים שבהם מותך הפלסטיק. לכל תנור מחוברים שני מדי טמפרטורה. מד טמפרטורה א׳ מודד את הטמפרטורה החיצונית (במעטפת התנור) ומד טמפרטורה ב׳ מודד את הטמפרטורה הפנימית (בגוף התנור). מערכת הבקרה נועדה לפיקוח על הפרשי הטמפרטורה בין גוף התנור לבין המעטפת שלו כך שהטמפרטורה הנמדדת בגוף התנור תהיה תמיד גבוהה לפחות ב-50 מעלות צלזיוס מן הטמפרטורה שבמעטפת התנור. אם ההפרש בין הטמפרטורות בין גוף התנור לבין המעטפת שלו הוא 50 מעלות צלזיוס או פחות, יש להוציא אות להפעלת החימום ואות להפסקת הקירור, ואם ההפרש ביניהן גדול מ-50 מעלות צלזיוס, יש להוציא אות להפסקת החימום ואות להפעלת הקירור. הניחו שהטמפרטורה שמודד מד הטמפרטורה ב׳ תהיה תמיד גבוהה מהטמפרטורה שמודד מד הטמפרטורה א׳, וכי הטמפרטורות הנמדדות הן בטווח שבין 0 מעלות צלזיוס ל-200 מעלות צלזיוס.

בנו תרשים זרימה לתכנית המדמה את הפעולה של מערכת הבקרה. הניחו שהטמפרטורה הנמדדת על-ידי מד הטמפרטורה א׳ נשמרת בזיכרון בתא בשם Temp_Out, והטמפרטורה הנמדדת על-ידי מד הטמפרטורה ב׳ נשמרת בזיכרון בתא בשם Temp_In. כמו כן, הניחו כי האות '1' מפעיל את החימום ואת הקירור, והאות '0' מפסיק את החימום ואת הקירור.

פתרון

נבחן מספר דוגמאות לטמפרטורות שנמדדו בתנורים (כל המעלות נמדדות בצלזיוס):

טבלה 1

הפעולות במערכת של בקרת הטמפרטורות בתנור בשתי מדידות נבחרות

המקרה הנבדק	מד טמפרטורה א׳ (במעלות צלזיוס)	מד טמפרטורה ב׳ (במעלות צלזיוס)	הפרש טמפרטורות בין מד טמפרטורה ב׳ למד טמפרטורה א׳ (במעלות צלזיוס)	הפעולות שיש לבצע
א׳	110	180	70	הפסק חימום
ב׳	125	160	35	הפעל חימום

נבצע את בדיקת התוצאות של מדידות הטמפרטורה בשני שלבים:

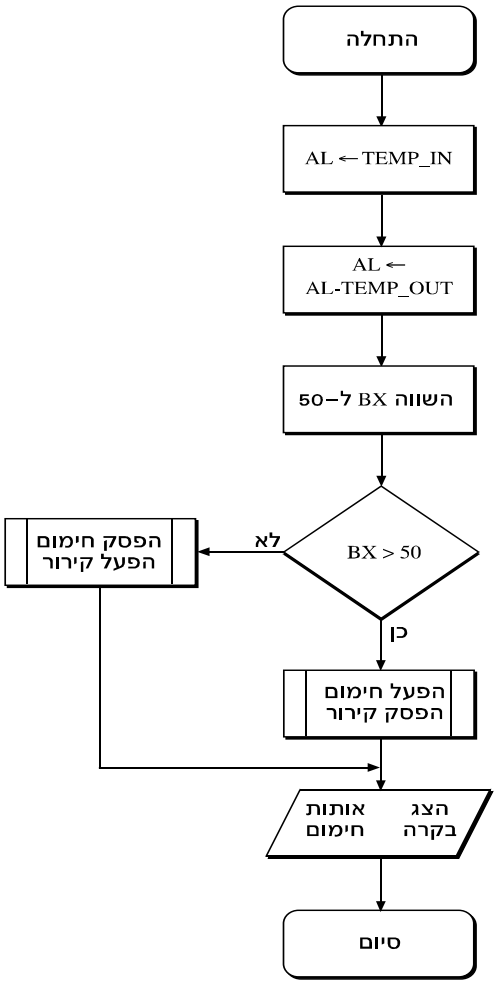
נפחית: Temp_OUT –Temp_IN

1. אם התוצאה המתקבלת גבוהה מ-50 – אזי יש להפסיק את החימום ולהפעיל את הקירור.
2. אם התוצאה המתקבלת היא 50 או פחות – אזי יש להפעיל את החימום ולהפסיק את הקירור.

שאלה: איזו פעולה מבין הפעולות ב-1 וב-2 תיעשה בכל אחד מן המקרים המוצגים בטבלה 1?

תשובה: במקרה א' תיעשה פעולה ב-1, ובמקרה ב' תיעשה פעולה ב-2.

להלן תרשים זרימה מתאים:



איור 2
תרשים זרימה לתכנית ביקורת על הטמפרטורות של תנורים

פתרונות של תרגילים נבחרים מפעילות 4

תרגיל 4

א.

```
.MODEL SMALL
.STACK  100h
.DATA
.CODE
start:
        mov  ax,   @DATA
        mov  ds,   ax

        mov  ch,   63
        mov  cl,    ch
        add   cl,    ch
        add   cl,    ch
        add   cl,    ch

        mov   ax,   4c00h
        int     21h
END start
```

ב. CH = 3Fh

ג. CH = 1Fh

פתרונות של תרגילים נבחרים מפעילות 5

תרגיל 4

אפשר לבדוק אם נתון מכיל סיבית אחת לפחות פעם אחת בעזרת ההוראות:

```
mov  al,  12                    ; הנתון הנבדק
cmp  cl,  0
```

תרגיל 6

התכנית מבצעת את הבדיקה הבאה:

```
אם cl ≠ char1 וגם cl = char2 אזי
        check =1
אחרת
        check = 0
```

פתרונות של תרגילים נבחרים מפעילות 6

תרגיל 1

א. התכנית אינה מטפלת במקרה ש-y = 0. הסיבה – בזמן ביצוע ההוראה LOOP מעודכן תחילה ערכו של מונה הלולאה CX (קטן באחת) ואחר כך נבדק תוכנו. ערכו של CX יהיה FFFF והלולאה תתבצע עד שהוא יתאפס. כדי לפתור את הבעיה יש להוסיף הוראות לבדיקת ערכו של y.

ב.

```
.MODEL SMALL
.STACK 100h
.DATA
        x  dw  15
        y  dw  50

.CODE
start:
    mov  ax,  @data
    mov  ds,  ax
    mov  ax,  0                    ; מסכם
    mov  cx,  y                    ; מונה
    cmp  cx,  0                    ; אם כופל הוא 0
```

```
    jz  sof                                   ; לולאת כפל
   again:  adc  ax,  x
           loop again
sof :
                                              ; סיום התכנית
    mov  ax,  4c00h
    int    21h
END start
```

תרגיל 4

(1) jns
(2) jz
(3) jg
(4) jz
(5) dh

תרגיל 5

```
.MODEL SMALL
.Stack 100h
.DATA
        num  DW  4
        sum  DW  0
.CODE
start:
                                              ; אתחול מקטע נתונים
        mov  ax,  @DATA
        mov  ds,  ax
                                              ; אתחול משתנים
        mov  dx,                              ; איבר המקודם בגוף לולאה
        mov  cx,  1                           ; cx מונה לולאה חיצונית
doloop1:        mov  bx,  cx                  ; bx מונה לולאה פנימית
```

```
doloop2:    inc  bx                    ; bx = bx+1
            cmp  bx,  num              ; ? num = bx האם
            jae  endloop2
            inc  dx
            jmp  doloop2
endloop2:                              ; סיום לולאה פנימית
            cmp  cx,  num              ;? num = cx האם
            inc  cx
            jmp doloop1
endloop1:                              ; סיום לולאה פנימית
mov  sum,  ax
mov  ax,  4C00h
int    21h
END start
```

פתרונות של תרגילים נבחרים מפעילות 7

תרגיל 2

המשתנה books מכיל את מספרי הספרים שקראו. כל סיבית (על פי מיקומה במספר) '1' מציינת כי ספר נקרא וסיבית '0' מציינת שספר לא נקרא.

לדוגמה 0000000101110110b מציין כי ספרים 2, 3, 5, 6, 7 ו-9 נקראו.

```
.MODEL SMALL
.Stack 100h
.DATA
        books       DW  0000000101110110b
        result1     DB   0
        result2     DB   0
        result3     DB   0
        result4     DB   0

.CODE
```

```
start:
        mov  ax, @DATA
        mov  ds, ax

        test  books, 10000b                    ; האם קרא ספר מספר 5
        jz  next1
        inc  result1
                                               ; הקורא קרא ספר מספר 3 או ספר מספר 7?
next1:  test  books, 1000100b
        jz  next2
        inc  result2
                                               ; הקורא קרא את כל עשרת הספרים?
next2:  cmp  books, 1111111111b
        jnz  next3
        inc  result3
                                               ; האם הקורא לא קרא אף ספר?
next3:  cmp  books, 0
        jnz  finish
        inc  result4
finish:
        mov  ax, 4C00h
        int  21h
END start
```

פתרונות של תרגילים נבחרים מפעילות 8

תרגיל 8

התכנית מוצאת כמה פעמים שסיבית '0' מופיעה באוגרים al ו-ah באותם המקומות. התוצאה תהיה באוגר dl. לדוגמה אם נתון כי:

```
al = 01001011b
ah = 11011010b
```

בסיום התכנית ערכו של dl הוא 2.

הפתרון מבוסס על הרעיון הבא: הפיכת תוכן האוגרים al ו-ah ומיסוך al באמצעות ah ולסיום, ספירה של מספר הסיביות שהן '1' באותו מיקום בשני האוגרים al ו-ah.

תרגיל 11

מימוש שתי הפעולות המתוארות באלגוריתם באמצעות הוראות לוגיות:

א. לחיבור הספרות של שני המספרים נשתמש בפעולה: A XOR B

ב. לטיפול בנשא נשתמש בפעולה: SHL (A AND B)

```
.Stack 100h
.DATA
    a       DW  29h
    b       DW  15h

.CODE
start:
    mov  ax, @DATA
    mov  ds, ax

again:  mov  ax, a
        mov  bx, b
        xor  ax, bx          ; חיבור ספרות
        mov  dx, ax
        mov  ax,  a
        and  ax,  bx         ; טיפול בנשא
        shl  ax,  1
        mov  a,   dx         ; עדכון הכופל והנכפל
        mov  b,   ax
        cmp  ax,  0          ; בדיקה האם יש נשא
        jz   finish
        jmp  again
```

```
finish:
        mov  ax,  4C00h
        int     21h
END start
```

תרגיל 12

תכנון פתרון: ייצוג כסף בכפולות של 10 א"ג, למשל 3.20 ₪ כ-320 א"ג

הפעולות שיש לבצע הן:

```
coins – sum = change
m = change div 50
r1 = change mod 50
r2 =  r1 div 10
```

התכנית:

```
.MODEL  SMALL
.STACK 100h
.DATA
        ten     DB  10
        fifty   DB  50
        sum     DW  680
        coins   DW  1000
        x1      DB   ?
        x2      db   ?
.code
start:
   mov  ax,    @DATA
   mov  ds,    ax

        mov  ax,  coins                         ; ax = coins
        sub  ax,  sum
        div  fifty                  ; al = sum div fifty, ah=sum mod fifty
        mov  x1,  al                            ; x1 = al number of fiftys
```

```
        mov  al,  ah                    ; ax = ah = sum – x1·fifty
        mov  ah,  0
        div   ten                       ; al = ax div ten, al mod ten
        mov  x2,  al                    ; 2= number of tens

   mov  ax,  4c00h
   int  21h
END start
```

תרגיל נוסף:

התכנית הבאה מכפילה שני מספרים מטיפוס בית ללא סימן וללא שימוש בהוראת כפל.

```
.MODEL SMALL
.STACK  100h
.DATA
.CODE
start:
        mov  ax,  @DATA
        mov  ds,   ax
        mov  ax,   5
        mov  bx,   7
     xor    DX,DX                   ; DX := 0 (keeps mult. result)
     mov  CX,7                      ; CX := # of shifts required
     mov   SI,AX                    ;  save original number in SI
repeat1:                            ; multiply loop – iterates 7 times
     rol    BL,1                    ; test bits of number 2 from left
     jnc    skip1                   ; if 0, do nothing
     mov   AX,SI                    ; else, AX := number1·bit weight
     shl    AX,CL
     add    DX,AX                   ; update running total in DX
skip1:
```

```
    loop    repeat1
    rol     BL,1                        ; test the rightmost bit of AL
    jnc     skip2                       ; if 0, do nothing
    add     DX,SI                       ; else, add number1
skip2:
    mov     AX,DX                       ; move final result into AX
            mov   ax,  4c00h
            int     21h
END start
```

פרק 6
שיטות מיעון, מערכים ורשומות

משך הלימוד

שש שעות לימוד עיוני, שש שעות התנסות

מטרות

- הכרת שיטות המיעון השונות
- הבנת ההשפעה של בחירה של שיטת מיעון מסוימת על מספר הגישות לזיכרון ועל משך ביצוע התכנית
- ההכרה של צורת המימוש וצורת העיבוד של מערך חד-ממדי, של מערך דו-ממדי ושל רשומה

נושאים עיקריים

- שיטות מיעון
- שיטות מיעון של זיכרון
- הגדרה של מערך חד-ממדי ושל מערך דו-ממדי
- הגדרה של רשומה
- בחירת שיטת מיעון מתאימה לעיבוד מערכים ורשומות

הידע שירכשו התלמידים בפרק זה יכשיר אותם:

- להעריך את מספר הגישות לזיכרון בתכנית המשתמשת בשיטת מיעון זיכרון ובתכנית המשתמשת בשיטת מיעון אוגר או במיעון מיידי.
- לממש תכנית המשתמשת בשיטת מיעון מסוימת.
- להגדיר מערכים חד-ממדיים ודו-ממדיים.
- להגדיר רשומות.
- לכתוב תכניות המשתמשות במערכים חד-ממדיים, דו-ממדיים וברשומות.

פעילויות לתלמיד

פעילות 9 – שיטות מיעון ומערכים

הרחבה

א. סיכום שיטות מיעון

טבלה הבאה מציגה שיטה שתקל עליכם לזכור את הצירופים האפשריים של שיטות מיעון הזיכרון במעבד 8086, כאשר שיטת מיעון יכולה להיות מורכבת: מאוגר בסיס ו/או אוגר אינדקס להם ניתן להוסיף גם העתק. בהתאם לכך ניתן להרכיב שיטת מיעון :

- מצירוף של איבר אחד בלבד המסומן בסוגריים מרובעים מעמודה 2 או 3 ;
- **או** לבחור איבר אחד המסומן בסוגריים מרובעים מעמודה 2 ו איבר אחד המסומן בסוגריים מרובעים מעמודה 3.
- לבחירה זו ניתן להוסיף העתק (בחירה של איבר מעמודה 1)

אוגר המקטע המצוין בסוגריים עגולים מתאר את המקטע אליו מתייחסת הכתובת המחושבת ואוגר מצביע מסומן בסוגריים מרובעים. כאשר יש צירוף של אוגר בסיס מעמודה שנייה עם אוגר בסיס מעמודה שלישית, מתייחסים למקטע אליו מתייחס אוגר הבסיס.

טבלה: מרכיבי שיטת המיעון

עמודה 1 – העתק	עמודה 2 – אוגר אינדקס	עמודה 3 – אוגר בסיס
(disp)	(DS:) [SI]	(DS:) [BX]
	(DS:) [DI]	(SS:) [BP]

נציג כעת כמה דוגמאות לרישום כתובת בשיטת מיעון זיכרון באמצעות הטבלה:

- בחירה בהעתק (disp) בלבד, מן העמודה הראשונה (בשמאל), מתייחסת למיעון ישיר ;
- בחירה בהעתק (מהעמודה הראשונה) ואוגר אינדקס SI (מהעמודה השלישית) מתייחסת למיעון אינדקס ישיר [SI+disp] במקטע הנתונים ;
- בחירה של אוגר הבסיס BX (מהעמודה השנייה) ואוגר אינדקס DI (מהעמודה השלישית) מתייחסת למיעון אינדקס-בסיס [BX+DI] במקטע הנתונים ;

– בחירה של disp (מהעמודה הראשונה), BP מהעמודה השנייה ו-SI מהעמודה השלישית מתייחסת למיעון אינדקס-בסיס עם היסט: [BP+SI+disp] במקטע המחסנית.

ב. רשומות

נציג דוגמה נוספת שבה נשתמש בשיטת מיעון זו כדי לעבד רשומה שאחד משדותיה הוא מערך, כדי לפתור את הבעיה הבאה:

במשרד הפקות עורכים מעקב על מספר הצופים בהצגה במשך שבוע. בכל שבוע נערכות שלוש הצגות ובסוף השבוע מחשבים כמה אנשים צפו בהן. הנתונים על ההצגה ועל מספר צופים בכל יום ויום נשמרים ברשומה מהמבנה הבא:

מספר הצגה מטיפוס מילה

מספר הצופים בכל הצגה נשמר במערך חד-ממדי בן שבעה תאים מטיפוס בית.

תחילה נתאר את מבנה הרשומה:

```
show struc
        code      DW ?
        visitors  DB ?
                  DB ?
                  DB ?
show ends
```

שימו לב! כדי להגדיר מערך כשדה ברשומה, יש להגדיר כל איבר בנפרד.

כדי לסכם את מספר הצופים בהצגה במשך שבוע נשתמש בשיטת מיעון אינדקס-בסיס עם העתק ונקבע כי:

- אוגר בסיס BX יצביע לתחילת הרשומה.
- אוגר אינדקס SI ישמש כמציין לאיבר במערך המוגדר בשדה visitors.
- העתק יציין את הכתובת האפקטיבית של השדה visitors מתחילת הרשומה.

נרשום אלגוריתם מתאים לסיכום איברי מערך הצופים:

```
                                    לכל i מ-1 עד DAYS בצע
                                sum = sum + s1.visitors[i]
```

ולהלן התכנית המתאימה:

```
.MODEL  SMALL
.STACK
DAYS EQU 3                          ; קבוע המציין את מספר ההצגות בשבוע
                                    ; רשומת הצגה
show struc
        code     DW ?               ; קוד ההצגה
        visitors  DB ?              ; מערך של מספר צופים בכל יום שבו נערכה הצגה
                 DB ?
                 DB ?

show ends

.DATA
s1 show<1234, 50,200,100>           ; אתחול רשומת הצגה שהקוד שלה הוא 1234
sum dw 0                            ; סכום מספר הצופים במשך השבוע

.CODE
start:
                                    ; אתחול מקטע הנתונים
   MOV      AX,   @DATA
   MOV      DS,   AX
                                    ; אתחול משתנים
LEA   BX,  s1                       ; מצביע לתחילת הרשומה
XOR   AX,  AX                       ; מספר הצופים בהצגה אחת
XOR   DX,  DX                       ; מספר הצופים הכללי
XOR   SI,SI                         ; אינדקס לאיבר במערך הצופים
MOV  CX,  DAYS                      ; מונה לולאה
again:
        MOV  AL, [BX+2+SI]          ; AL= s1.visitors[i]
        ADD  DX,  AX                ; סיכום מספר צופים
```

```
        INC    SI                              ; קידום מציין
        LOOP again
MOV  sum,  DX
                                               ; יציאה מתכנית
   MOV      AX,  4c00h
   INT      21h
END start
```

פתרונות של תרגילים נבחרים מפעילות 9

תרגיל 5

השגיאה היא השימוש בהוראה: mov [si + cx], al

להלן התכנית הנכונה:

```
.MODEL SMALL
.STACK 100h
.DATA
        place DB 11h
        buffer   db 10 dup(?)
.CODE
start:
        mov  ax,  @DATA
        mov  ds,  ax
        mov  al,   byte ptr Place
        lea    si,   buffer
        mov  cx,  10
again:  mov  [si],  al
        inc    si
        loop  again
        mov  ax,  4c00h
        int    21h
END  start
```

תרגיל 6

נרחיב את תרגיל 6 כך שיטפל בציונים בין 0 ל-100.

תכנון הפתרון:

א. כדי להשתמש במערך מונים בין 0 ל-10 נחלק תחילה את הציון ב-10.

ב. בתכנית נשתמש בשני מערכים:

- מערך ציונים מטיפוס בית עליו מצביע משתנה marks
- מערך מונים מאופס מטיפוס בית עליו מצביע counter

ג. שיטות המיעון שבהן נשתמש:

- לגישה לאיבר במערך mark – נשתמש בשיטת מיעון אוגר עקיף (באמצעות bx)
- לגישה לאיבר במערך counter – נשתמש בשיטת מיעון אינדקס (באמצעות אוגר si)

התכנית:

```
.MODEL SMALL
.STACK 100h

.DATA
        marks   DB      60, 70, 50, 70, 80, 80, 90, –1
        counter DB 11 dup(0)
.CODE
start:
        start:
        mov  ax,  @DATA
        mov  ds,  ax
        mov  ax,  0
        mov  cl,  10
        lea     bx, marks
again:  mov     al,  [bx]
        cmp   al, –1
        je  finish
        div   cl
```

```
        and  ax,  0Fh
        mov  si,  ax
        inc   counter[si–1]
        inc   bx
        jmp   again

finish:  mov  ax,  4c00h
        int    21h
END  start
```

תרגיל 14

```
.MODEL SMALL
.STACK 100h
```

הגדרה של מבנה רשומת רץ ;

```
runner  STRUC
        code    DB   ?
        score   DB   ?
runner  ENDS
N EQU 5
```

מספר המתחרים ;

```
.DATA
        winner  db  0
```

המספר הזוכה בתחרות ;

אתחול הערכים של חמש רשומות ;

```
        p1      runner <11, 154>
        p2      runner <43, 149>
        p3      runner <25, 156>
        p4      runner <9, 151>
        p5      runner <15, 157>

.CODE
start:
```

```
                                        ; אתחול מקטע הנתונים
        mov   ax,  @DATA
        mov   ds,  ax

        lea     bx, p1
        inc     bx
                                        ; קידום מצביע לרשומה הבאה
        inc     bx
        cmp    al, [bx+1]               ; השוואה של תוצאת ריצה של הרץ הבא
        jae  nextrec
        mov  ah, [bx]                   ; עדכון נתוני הרץ הטוב ביותר
        mov  al, [bx+1]
nextrec: loop  again
        mov winner, ah                  ; שמירה על מספר הרץ הטוב ביותר
                                        ; יציאה מהתכנית
 mov  ax, 004ch
 int    21h
END  start
```

פרק 7
מחסנית, שגרות ומקרו

משך הלימוד

שש שעות לימוד עיוני, שש שעות התנסות

מטרות

- הבנת התפקיד ואופן השימוש במחסנית לפתרון בעיות
- הכרת מנגנון המימוש של נהלים, העברת פרמטרים ומשתנים מקומיים
- הכרת שיטות להעברת פרמטרים לפי ערך ולפי כתובת
- הכרת אופן המימוש של פונקציה
- הבנת אופן הביצוע של פונקציה רקורסיבית פשוטה
- הבנת מנגנון המקרו
- יכולת כתיבה של מקרו פשוט

נושאים עיקריים

- מחסנית
- הוראות לדחיפת איברים למחסנית ולשליפת איברים מהמחסנית
- פרוצדורה
- פרמטר לפי ערך, ופרמטר לפי כתובת
- משתנים מקומיים
- פונקציה
- רקורסיה
- מקרו

הידע שירכשו התלמידים בפרק זה יכשיר אותם:

- לכתוב תכניות המשתמשות בהוראות מחסנית.
- לכתוב, לזמן ולפענח פרוצדורות.
- להעביר פרמטרים לפי ערך ולפי כתובת.
- להקצות משתנים מקומיים ולהשתמש בהם.
- לכתוב, לזמן ולפענח פונקציה.
- לפענח פונקציה רקורסיבית.
- לכתוב פונקציה רקורסיבית פשוטה.
- לכתוב, לזמן ולפענח מקרו פשוט.

פעילויות לתלמיד

פעילות 10 – מחסנית, פרוצדורות ומקרו

הרחבה

א. רקורסיה – חישוב סדרת פיבונצ'י

בסדרת פיבונצ'י כל איבר הוא סכום של שני האיברים הקודמים לו.

הגדרה רקורסיבית לסדרת פיבונאצ'י:

שני האיברים הראשונים הם (לפי ההגדרה):

$fib(1)=1$, $n=1$

$fib(2)=2$, $n=2$

האיבר הכללי עבור $n>2$ הוא

$fib(n) = Fib(n-1)+fib(n-2)$, $n>2$

אלגוריתם:

```
אם n=1 אזי
    fib = 1
אחרת אם n=2 אזי
    fib = 2
אחרת
fib = fib(n–1) + fib(n-2)
```

בתכנית יש שתי קריאות רקורסיביות: האחת לחישוב fib(n-1) והשניה לחישוב fib(n-2).
n מאוחסן ב-BX ו-AX מחזיר את תוצאה

```
.MODEL  SMALL
.STACK  100h
.DATA
.CODE

fib PROC
        CMP  BX,  1                     ; if BX=1 ?
        JNE try_two
        MOV  AX,  1                     ; if n–1 then fib=1
        MOV  AX,  1
        JMP done
try_two: CMP  BX,  2
        JNE  call_fib
        MOV  AX,  2                     ; else if n=2  then fib=2
        JMP done
```

```
call_fib: DEC  BX                    ; else call fib(n-1)
          PUSH BX                    ; להעברת מונה B לקריאה רקורסיבית הבאה
          CALL  FIB
          POP   BX
          DEC   BX                   ; BX = n–2
          PUSH AX                    ; שמירת ערך במחסנית fib(n–1)
          CALL FIB                   ; CALL fib(n–2)
          POP   DX                   ; DX = fib(n–1)
          ADD   AX,  DX              ; AX = fib(n–2) + fib(n–1)
done:     ret                        ; חזרה
ENDP fib

start:
   MOV     AX,    @DATA
   MOV     DS,    AX

   MOV     BX,    4                  ; BX = 4
   CALL  fib                         ; factorial(4)
exit:
   MOV     AX,    4ch
   INT       21h
END  start
```

ב. פונקציה רקורסיבית המקבלת מספר כלשהו n, ומכניסה למערך (שאורכו n) את כל המספרים מאחד עד n

```
.model  small
.stack  100h
.data
```

```
n   dw  008h                    ; n מספר כלשהו
arr db  10 dup('0')             ; הגדרת מערך שאורכו קטן שווה n

.code
start:
        mov  ax,  @data
        mov  ds,  ax

        mov  SI,  offset n      ; הכנסת המשתנה n לאוגר BX
        mov  BX, [SI]
        mov   si,  offset arr   ; הכנסת כתובת המערך למשתנה Si
        call    oneton          ; קריאה לפונקציה
sof:  mov  ax,  4c00h
      int    21h

oneton  PROC
        cmp  bx,  00h           ; תנאי עצירה
        je done
        push  bx                ; הכנסת המשתנה למחסנית
        dec    bx               ; הקטנת המשתנה באחד
        call    oneton          ; קריאה חוזרת לפונקציה
        pop    bx               ; הוצאת המשתנה מהמחסנית
        mov   [si],  bl         ; הכנסת המשתנה למערך
        inc    si
       done:  ret
  ENDP oneton
END start
```

ג. מקרו המשנה אות אנגלית קטנה לאות גדולה:

הגדרת המקרו:

```
toUpper     MACRO ch
            LOCAL done
            cmp ch, 'a'
            jb done
            cmp ch, 'z'
            ja done
            sub ch, 32
done:
ENDM
```

דוגמה לזימון המקרו:

```
mov al, 'b'
toUpper al
mov bl, al
mov ah, '1'
toUpper ah
mov bh, ah
```

שאלה למחשבה:

הסבירו מדוע השתמשנו בהצהרה LOCAL?

פרק 8
פסיקות וקלט/פלט

משך הלימוד

שש שעות לימוד עיוני, שש שעות התנסות

מטרות

- הבנת התפקיד של פסיקה ואופן ביצועה
- לימוד השימוש בפסיקות שירות לביצוע פעולות קלט/פלט
- הבנה של כתיבת פסיקה

נושאים עיקריים

- פסיקה
- סוגי פסיקות
- נהלי שירות 21h
- כתיבת תכנית המבצעת פעולות קלט /פלט
- טבלת פסיקות

הידע שירכשו התלמידים בפרק זה יכשיר אותם:

- לכתוב תכניות המשתמשות בפסיקות שירות 21h לביצוע פעולות קלט/פלט
- להכיר סוגים שונים של פסיקות
- לדעת להשתמש בטבלת פסיקות
- לכתוב פסיקה פשוטה

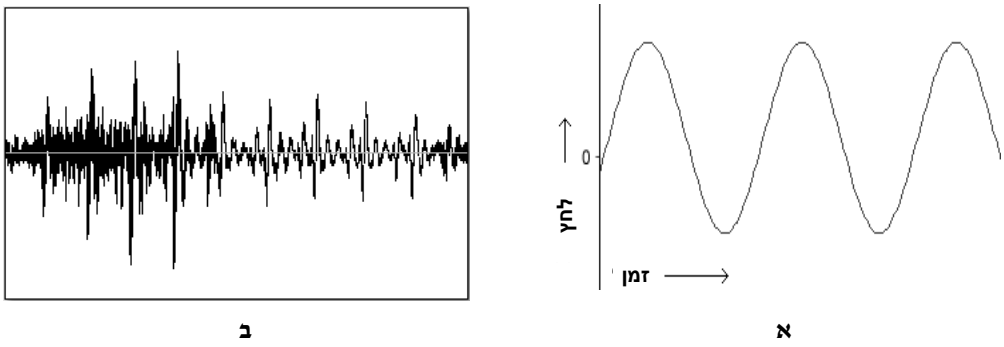
פעילויות לתלמיד

פעילות 11 – פסיקות וקלט /פלט

הרחבה

א. שימוש בפסיקות BIOS להשמעת קול

קול נוצר כאשר גוף מתנודד מהר וכתוצאה מתנודתו נוצרים הבדלי לחצים באוויר. הפרשי הלחצים האלה נקלטים באוזן ומתורגמים על-ידי מוח האדם לקול. גלי קול הם גלים תקביליים (אנלוגיים) המתארים השתנות רציפה של לחץ האוויר כתלות בזמן. באיור 3 (בחלק א') רשום גרף אשר מציג קול המורכב מכמה גלי קול. בחלק ב' של האיור מוצג גל יחיד המבטא השתנות של לחץ כתלות בזמן.



א ב

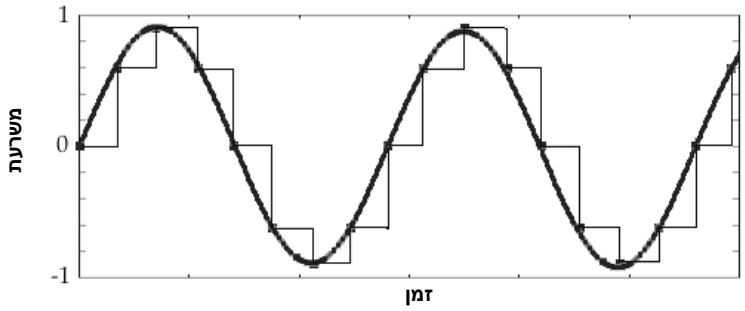
איור 3
א. תיאור של גלי קול
ב. משרעת (אמפליטודה) של גל קול כתלות בזמן

גל תקבילי מתואר על-ידי שני פרמטרים: גובה הגל, הנקרא משרעת, ותדירות הקצב המציינת את השתנות הגל.

- **המשרעת** מתארת את עוצמת הלחץ בכל נקודת זמן והיא נמדדת בדציבלים. ככל שהמשרעת גבוהה יותר הצליל נשמע חזק יותר. טווח השמיעה של האדם הוא בין עשרה ל-150 דציבלים. דציבל הוא עשירית בל (bel) שזו יחידת מידה המשמשת להשוואה בין שתי רמות הספק. גידול בבל אחד פירושו עוצמה חזקה פי עשרה.

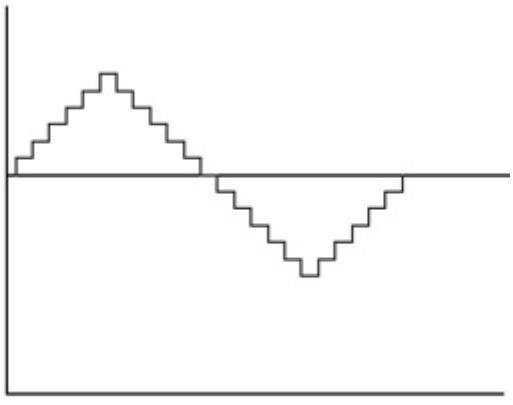
- **תדירות** היא קצב ההחזרה של גל קול כלשהו בזמן. במילים אחרות, כמה פעמים צורה מסוימת של משרעת חוזרת במהלך שנייה אחת. תדירות הגל קובעת את גובה הצליל. גל בתדירות גבוהה נשמע לאוזן האדם כצליל גבוה (לדוגמה צחוק של ילד), וגל בתדירות נמוכה נשמע כצליל נמוך (דוגמה: צליל של באס). יחידת המדידה של התדירות היא הרץ (Hz). לדוגמה, בגל שתדירותו היא 880 הרץ, יש 880 מחזורים (חזרות) של הגל בשנייה אחת. תחום התדרים שהאדם מסוגל לשמוע הוא בין 20 ל-20,000 הרץ.

כדי לשמור מידע על קול במחשב, יש להמיר את הרישום של גלי הקול התקביליים למידע סִפרתי. לשם כך דוגמים את גל הקול בכמה פרקי זמן, וממירים את עוצמת הקול שנמדדה לקוד בינארי. באיור 4 מוצג גל תקבילי של סינוס ועליו מסומנות בשחור הנקודות שנדגמו. במחשב מאוחסנים מספרים בינאריים בני שלוש סיביות המציינים את גודל המשרעת בכל דגימה.



איור 4
דגימת גל קול וייצוג דגימה באמצעות שלוש סיביות

אילו דגמנו בקצב כפול, היינו מקבלים קירוב טוב יותר של הגל הסִפרתי, אבל אז היה עלינו להשתמש במספר סיביות רב יותר, כמתואר באיור 5. כדי לאחסן את המידע של דגימות הגל שבו יש 16 ערכים שונים, נצטרך להשתמש בארבע סיביות.



איור 5
דגימת גל קול וייצוג דגימה באמצעות ארבע סיביות

שני פרמטרים משפיעים על איכות הקול שנשמר במחשב: מספר הדגימות (או פרק הזמן שבין שתי דגימות עוקבות) ומספר הסיביות שבהן משתמשים לשמירת המידע על כל דגימה. ככל שפרקי הזמן שבהם הקול נדגם קצרים יותר המידע הנשמר על כל דגימה מכיל סיביות רבות יותר וכך הוא מאפשר לשמור מידע מדויק יותר על עוצמת הקול. ככל שמספר הדגימות גדל, איכות הקול הנשמרת במחשב גבוהה יותר ובמקרה כזה נוכל לומר כי ייצוג הקול משמר מידע ברמה המספקת את הדרישות. מבחינה מעשית אי-אפשר להמיר את כל רצף הנקודות בגל תקבילי למידע סְפרתי משום שמדידה כזו היא אינסופית ולכן המרה זו אינה משמרת מידע. ואף-על-פי-כן, כמות המידע הנדגמת והמיוצגת במחשב מספיקה כדי להפיק צלילים הנשמעים לאוזן באיכות סבירה, והרי זו אמת המידה הקובעת.

כאמור, כדי להפיק קול ממחשב יש להמיר את המידע הסְפרתי לגל תקבילי המשמש כקלט לרמקול. הרמקול כולל בתוכו מֶמברנה (קרום דק) המתנודדת בהתאם לתנודות הגל התקבילי ויוצרת הפרשי לחץ ובמילים אחרות – יוצרת גלי קול.

שאלה למחשבה

דגמו קול בקצב של 64 פעמים בכל מחזור וחשבו כמה סיביות דרושות כדי לייצג דגימה אחת.

פרוצדורות שירות להשמעת קול

שימוש בפרוצדורות שירות מאפשר להציג קול באמצעות הרמקול הפנימי המותקן במחשב האישי. כפי שתואר בפרק השני, קול נוצר כאשר גוף מתנודד מהר וכתוצאה מתנודתו נוצר גל המתקדם בחומר שהגוף נמצא בו. במחשב אפשר ליצור קול על-ידי יצירת תנודות המניעות את הממברנה של הרמקול הפנימי בתדרים שאוזן אדם עשויה לקלוט.

את התנודות אנו יכולים ליצור בעזרת השעון הפנימי של המחשב. בכל מחשב יש שעון פנימי המגדיר את קצב העבודה של המעבד. השעון הפנימי מפיק תנודות של גל ריבועי. במעבדים מתקדמים תדר השעון הפנימי הוא גדול מאוד. תדר זה גבוה מאוד ואינו נקלט באוזן של אדם. לכן, כדי לייצר תדרים שאוזן אדם יכולה לשמוע, יש להקטין תחילה את תדר השעון. כלומר, התכנית צריכה לפנות לשעון הפנימי ולבקש תדרים מסוימים.

תדר של שעון PC הוא 1.193186 MHz והוא מופק במפתח (port) שמספרו 42h. כדי להשמיע קול, במפתח שמספרו 42h, המפיק את תדר השעון, צריך לקבל ערך של 16 סיביות שהוא מחלק את תדר השעון וכך מקטינו לתחום תדרי השמע. הטבלה הבאה מציגה שמונה אוקטבות של תווים מוזיקליים.

טבלה 2
אוקטבות (5 ו-6) של טונים מוזיקליים

טון	אוקטבה 5	אוקטבה 6
C	261.6	523.2
C#	277.2	554.4
D	293.7	587.3
D#	311.1	622.2
E	329.6	659.3
F	249.2	698.5
F#	370.0	740.0
G	392.0	784.0
G#	415.3	830.6
A	440.0	880.0
A#	466.2	932.3
B	493.9	987.8

שימו לב שהתדר של כל תו מכפיל את עצמו באוקטבה הבאה. MIDDLE C בפסנתר התואם לתו C באוקטבה 5 הוא התדר 261.6Hz. כדי לתכנת את השעון לתדר זה, יש לחשב את המחלק:

$$1,193,186 : 261.6 = 4561 = 11D1h$$

כלומר, יש לחלק את תדר השעון ב-11D1h כדי לקבל תו בתדר 261.6Hz.

כדי להשמיע קול ברמקול הפנימי על התכנית:

1. להורות לערוץ 2 של השעון לתפקד כמתנד של גל ריבועי. זאת נעשה על-ידי שליחה של הנתון B6h למפתח 43h.
2. להוציא מחלק לתדר (תחילה לבית תחתון ואחר כך לבית עליון) למפתח 42h (בדוגמה שלנו קודם D1h ואח"כ 11h).
3. להפעיל את הרמקול על-ידי שליחת ביטים '0' ו-'1' ל-PPI port 61h. שימו לב, ביטים 2-7 של מפתח זה משויכים לפונקציות אחרות ולכן אסור לשנות את ערכם.

אלגוריתם:

```
אתחול נתונים:
    SI מצביע על טבלת התווים
    דגל הכיוון מצביע בכיוון למעלה
לכל מקש שנלחץ נגן את התו הבא
    המתן ללחיצת מקש – פסיקת שרות 16h
    קרא את התו הבא  AX = musictabel[SI]
    אם זה לא הטבלה  AX ≠ 0  אזי
        השמע את הטון: הפעל גל ריבועי
        העבר את המחלק (ערך הנקרא מהטבלה)
        אפשר רמקול

סיים את התכנית
```

התכנית משמיעה תו אחד בכל לחיצת מקש עד שהערך בטבלת התווים הוא 0000.

```
; music.asm
.MODEL  SMALL
.STACK 100h
.DATA
                musictable      DW  11d1h, 0FDFh, 0E24h, 0D59h, 0BE4h,
                                    0A98h, 0970h, 08E9h, 0000h
.CODE
```

```
start:
                                        ; אתחול מקטע הנתונים
   mov  ax,  @DATA
   mov  ds,  ax
        mov  si,   musictable
        cld

II:     mov  ah,  0
        int    16h

        lodsw
        mov  bx,  ax
        cmp  ax,   0
        jz  done

        mov al, 0b6h
        out   43h, al

        mov  al,  bl
        out    042h,  al

        mov al, bh
        out   042h,  al

        in   al,     61h
        or   al,     3
        out  61h,  al
        jmp  II
done:
        in     al,     61h
        and  al,     1Ch
```

```
        out  61h,  al
        int  20h

                                                    יציאה מתכנית ;
   mov      ax,     4c00h
   int      21h
 END  start
```

ב. שימוש בפרוצדורות שירות מספר 10h להצגת פיסקל

התכנית הבאה מדגימה שימוש בפרוצדורות שירות לייצוג פיקסל. התכנית מציגה ארבעה פיקסלים. לחיצה על מקש מסיימת את התכנית והבקרה חוזרת למערכת ההפעלה.

```
.model small
.STACK 100h
.DATA
.CODE

start:
MOV  AX,  @DATA
MOV  DS,  Ax
   mov  ax,  13                                  ; mode = 13h
   int    10h                                   ; call bios service

   mov  ah,  0Ch                                 ; function 0Ch
   mov  al,   4                                  ; color 4 – red
   mov  cx,  160                                 ; x position = 160
   mov  dx,  100                                 ; y position = 100
   int  10h                                      ; call BIOS service

   inc  dx                                       ; plot pixel downwards
   inc  dx                                       ; plot pixel downwards
```

```
int   10h                          ;call BIOS service
inc   cx                           ;plot pixel to right
inc   cx                           ; plot pixel to right
int   10h                          ; call BIOS service
dec  dx                            ; plot pixel up
dec  dx                            ; plot pixel up
int   10h                          ; call BIOS service

xor    ax,    ax                   ; function 00h – get a key
int    16h                         ; call BIOS service
mov  ax,  3                        ; mode = 3
int    10h                         ; call BIOS service

mov  ax,  4C00h                    ; exit to DOS
int    21h

end  start
```

ג. שימוש במודול המטפל בפעולות קלט/פלט

פיתוח תכנית היא משימה מורכבת מאוד ואחת הדרכים להתמודד אתה היא לחלקה למשימות משנה הממומשות כפרוצדורות. אולם שימוש חוזר בפרוצדורה אפשרי רק באותה תכנית. אם ברצוננו להשתמש באותן פרוצדורות בתכנית אחרת עלינו להעתיקן לאותה תכנית. שיטה זו אינה נוחה ולעיתים גם יוצרת בעיות. אחת הבעיות היא שבמקרה ששינינו את הפרוצדורה, עלינו לעבור על כל התכניות שבהן השתמשנו בה ולבצע כל שינוי בנפרד. כדי להתמודד עם בעיות אלו אפשר לבנות תכנית ממספר מודולים באופן כזה, שכל מודול יטפל בחלק מהמשימה ובסיום נרכיב את התכנית מאוסף כל המודולים.

המודול, בניגוד לפרוצדורה, הוא קובץ נפרש שאפשר להדר אותו ולאחר מכן לקשר אותו לתכנית עצמה. כך, למשל, אפשר ליצור מודול המטפל בפעולות קלט/פלט ולהשתמש בו בתכניות שונות. בסעיף זה נציג כיצד אפשר לבנות מודול ולקשרו לתכנית המשתמשת בפרוצדורות שהוגדרו בו.

המודול שנבנה יכיל פרוצדורות קלט/פלט המוצגות בסעיף זה והן:
printChar, printString, getString. כמובן, שבאפשרותכם להוסיף פרוצדורות אחרות. בכתיבת הפרוצדורות יש להקפיד על תיעוד מתאים של הממשק, כך שמתכנת אחר שירצה להשתמש בהן יוכל להבין מה הוא מבצע וכיצד אפשר לקשור פרוצדורות אלה לתכנית שלו.

מבנה מודול קלט/פלט

מודול זה הוא בעל מבנה של תכנית בשפת סף. מאחר שאיננו מפעילים בו את הפרוצדורות אלא רק מגדירים אותן איננו מגדירים מחסנית. אנו יכולים להגדיר מקטע נתונים ולהגדיר משתני עזר ובמקטע הוראות לרשום את ההוראות לביצוע. כדי שתכניות אחרות יוכלו להפעיל את הפרוצדורות עלינו להגדיר את ההנחיה PUBLIC ואחר כך להגדיר רשימה של פרוצדורות שאפשר יהיה לזמן מתכנית אחרת. לדוגמה,

```
PUBLIC printChar,  printString, getString
```

בפרוצדורות עצמן הכנסנו שינוי אחד – שכל פרוצדורה שומרת את האוגרים שבהן היא משתמשת במחסנית וזאת, לפני ביצוע הוראות הפרוצדורה. בסיומה היא משחזרת את ערכיהם. שינוי זה הכרחי כדי שהפרוצדורות יהיו כלליות ושיתאפשר להשתמש בהן בתכניות אחרות מבלי לגרום להן נזק ומבלי לשנות את הערכים של האוגרים שבהן התכנית משתמשת.

להלן מודול פרוצדורות קלט/פלט הנשמר כקובץ בשם in-out.asm:

```
;in-out.asm
.MODEL  SMALL
PUBLIC printChar,  printString, getString
.DATA
                                                משתני עזר ;
.CODE
start:

                פרוצדורה המקבלת תו כפרמטר לפי ערך ומציגה אותו על הצג ;
printChar PROC
```

```
        PUSH  DX                        ; שמירת ערכי האוגרים
        PUSH  AX
        MOV   BP,  SP
        MOV   DL,  [BP+2]               ; התו להצגה
        MOV   AH,  2                    ; פרוצדורת שירות 2
        INT     21h
        POP     AX                      ; שחזור ערכי אוגרים
        POP     DX
        RET     2
printChar  ENDP

        ; פרוצדורה המחזירה פרמטר לפי אזכור שהוא כתובת של מחרוזת שהיא קולטת מהמשתמש
getString PROC
        PUSH  DX                        ; שמירת ערכי האוגרים
        PUSH  AX
        MOV  BP,   SP
        MOV  DX,  [BP+2 ]               ;כתובת המשתנה שבו תאוחסן המחרוזת שתיקלט
        MOV  AH,  0Ah                   ; פרוצדורת שירות 0Ah
        INT     21h
        POP     AX                      ; שחזור ערכי אוגרים
        POP     DX
        RET     2
getString  ENDP

        ; פרוצדורה המקבלת כקלט פרמטר לפי אזכור שהוא כתובת של מחרוזת ומציגה את המחרוזת
printString PROC
        PUSH  DX                        ; שמירת ערכי האוגרים
        PUSH  AX
        MOV  BP,   SP
        MOV  DX,  [BP+2]  ;             ; כתובת המחרוזת להצגה
        MOV  AH,  9                     ; פרוצדורת שירות 9h
```

```
        INT   21h
        POP   AX                    ; שחזור ערכי אוגרים
        POP   DX
        RET  2
printString  ENDP

END  start
```

בסיום, נהדר את הקובץ כתכנית רגילה.

כעת נכתוב את הקובץ הראשי המשתמש בפרוצדורות אלו. כדי להשתמש בפרוצדורות עלינו להוסיף מתחת להנחיה CODE את ההנחיה

EXTRN getString:proc, printString:proc, printChar:proc

הנחיה זו מודיעה למהדר כי הפרוצדורות נמצאות בקובץ אחר. ללא הנחיה זו המהדר לא יזהה את שמות הפרוצדורות ויוציא הודעות שגיאה.

להלן התכנית הראשית שנשמרה כקובץ ששמו: "example.asm"

```
.MODEL  SMALL
.STACK  100h

.DATA
        outMessage1   DB "input string:",  0Dh, 0Ah, "$"     ; הודעות פלט
        outMessage2   DB "the string is:",  0Dh, 0Ah, "$"
        inpBuffer     DB 11                                  ; מחרוזת קלט
                      DB 0
                      DB 11 dup(0)

.CODE
EXTRN getString:proc, printString:proc, printChar:proc
```

```
start:
                                        ; אתחול מקטע הנתונים
    MOV   AX, @DATA
    MOV   DS, AX

                                        ; הצגת הודעת פלט ראשונה
    MOV    BX, offset outMessage1
    PUSH  BX
    CALL  printString

                                        ; קליטת המחרוזת
    MOV  BX, offset inpBuffer
    PUSH  BX
    CALL  getString

                                        ; הצגת הודעת פלט שנייה
    MOV  BX, offset outMessage2
    PUSH  BX
    CALL  printString

                                        ; הצגת המחרוזת שנקלטה
    XOR  CX, CX                         ; אתחול מספר התווים
    MOV  SI, offset inpBuffer+1
    LODSB
    MOV  CL, AL
    CLD                                 ; כיוון קריאת המחרוזת
                                        ; לולאה להצגת תווים
doloop:
        LODSB                           ; התו שיש להציג
        CBW
        PUSH  AX
```

```
CALL printChar                    ; קריאה לפרוצדורה המציגה תו
LOOP doloop

                                  ; יציאה מתכנית
  MOV     AX,    4c00h
  INT     21h
END  start
```

לאחר הידור example.asm נקשר את שתי התכניות ונרשום:

```
tlink example+in-out
```

לאחר הקישור ייווצר קובץ הרצה בשם example.exe.

תרגיל לדוגמה

א. הוסיפו לקובץ הפרוצדורות פרוצדורה בשם getChar שתקלוט תו מהמשתמש ותחזיר אותו.

ב. כתבו תכנית שתקלוט תווים מהמשתמש עד יוקש התו "$" ושתציג בסיום את כל התווים שנקלטו.

ד. פסיקה שוכנת זיכרון

התכנית הבאה מגדירה פסיקה שוכנת זיכרון (הממשיכה לפעול גם לאחר שהתכנית הסתיימה).

התכנית משמיעה צפצוף בביצוע כל פקודת DOS ומציגה הודעה כאשר תכנית כלשהי מתבצעת.

```
.286                                        ; just for Pusha and Popa commands
.model tiny
 codesg segment para
       assume  cs:codesg,ds:codesg,es:codesg     ; set all Segments to our program
       org   100h                            ; CS:100h, load our tsr to offset 100h

Begin:       jmp  Install                                    ; goto label install

  message   db   13,10,' TSR successfully loaded! ',13,10,'$'
  MSG       db   13,10,' Program Is Running! ',13,10,'$'
  OldInt21  dd   ?



;=========================================================
          ; when ever INT 21h is called,
          ; OS will hang running program and execute your routine,
          ; so you must save flags and some needed Registers of prevoius program

;=========================================================
           MyInt21:
      pushf                          ;                  save all flags onto stack
      pusha
      push    DS                     ; if not, dos can't execute running program
      CMP   ax,4b00h                 ;          if a program wants to be executed
      jne     continue

      mov     ax,0900h               ;                                 show message
      push    cs                     ;                                   CS=DS
      pop     ds                     ;           remark(;) this line and above line
      Lea     dx,byte ptr CS:MSG     ;          to see result on showing message!
      call    CallDOS                ;       call old int21 (see below describtion)
```

```
                                        ;       Warrning : if you wrote "Int 21h"
Continue:                               ; your "Myint21"  routine will execute
                                        ;      and every flags will be destroyed!
                                        ;    Result: = return address will be lost
    mov     ax,0E07h
    int 10h                             ;                                   BEEP
    pop     DS
    popa
    popf
    jmp     CS:OldInt21                 ;               Jump back to real int 21h
;=============================================================
CallDos     proc  Near
     pushf                              ;             always you must save flags
     call   dword ptr cs:OldInt21       ;             if you want call hooked int
     Retn                               ;               retn will popf then backs
CallDos     endp
;=============================================================
End_tsr:                                ;       all above code will be residented
Install:
    CLI                                 ; Disable interupts(CLear Interupt flag)
    push    cs  cs
    pop     ds  es                      ;                   be sure that CS=DS

    mov     ax,3521h                    ;                   get address int21h
    int     21h
    mov     word  ptr oldint21, bx
    mov     word  ptr oldInt21+2,es     ;                   Save interrupt address

    mov     ax,  2521h
    mov     dx,  offset cs:myInt21      ;       Modify the address of the int 21h
    int     21h                         ;                               with mine
```

```
    mov    ax,  0900h                ;                    show message
    lea    dx,byte ptr cs:message
    int    21h

    mov    ax,  3100h
    mov    dx,  offset End_tsr       ;                    Request for TSR

    STI                              ;       Enable interupts(SeT Interupt flag)

    int    21h                       ;                    notice! after this,
                                     ;                program will be shut down
                                     ; so we must "SeT Interupts" befor doing it
codesg  ends
    end    begin
```

פרק 9
מחרוזות

משך הלימוד

ארבע שעות לימוד עיוני, ארבע שעות התנסות

מטרות

- הכרת ההוראות לעיבוד מחרוזות
- יכולת לייצג מחרוזת בשפת אסמבלי
- כתיבת תכניות המשתמשות בעיבוד מחרוזות

נושאים עיקריים

- המבנה של הוראת מחרוזת
- דגל הכיוון
- שימוש במקטעי נתונים: DS ו-ES
- הוראות מחרוזת: MOVS, LODS, STOS, CMPS, SCAS, XLAT (המטפלות בבית ובמילה)
- שימוש באופרטורים לחזרה: REP,REPE, REPZ, REPNE, REPNZ

הידע שירכשו התלמידים בפרק זה יכשיר אותם:

– לכתיבת תכניות המשתמשות בהוראות מחרוזת לפתרון בעיות.

פעילויות לתלמיד

פעילות 12 – הוראות מחרוזת

הרחבה

כתבו תכנית הקוראת מחרוזת ומוסיפה תו רווח לפני כל תו "?". לדוגמה, מחרוזת המקור היא: "to be? or not to be?" לאחר השינוי תתקבל המחרוזת הבאה:
"to be ? or not to be ?".

```
.model  small
.stack
MAX_A EQU 20
MAX_B EQU 40
.data
     a  DB "to be? or not to be"?
     b  DB MAX_B dup(?)
.code
start:
          mov     ax,    @DATA
          mov     ds,    ax
          mov     es,     ax

               mov  si,  offset a
               mov  di,  offset b
               mov  cx,  MAX_A
               cld
again:
          lodsb                                   ; AL = a[SI]
          cmp al, '?'                             ; if AL = '?'
                                                  ; AL <> '?' jump cont
        jne cont
                                                  ; AL = '?'
               mov al,   ' '
               stosb
```

```
                mov  al,   '?'
          cont:
                stosb                                   ; b[DI] = AL
                loop again
sof:    mov     ax,    4c00h
        int     21h
END  start
```

פרק 10
ארכיטקטורה של מעבדים מתקדמים

משך הלימוד

שבע שעות לימוד עיוני

מטרות

- להציג את ההתפתחויות והשינויים שחלו במהלך השנים במבנה מחשבים
- לסקור ולהכיר את משפחת המעבדים 86× של אינטל ולהציג נקודות ציון המתייחסות לשיפורים בחומרה.
- להבין שיקולי תכנון של חומרה (ארכיטקטורת צינור הוראות ושיטות ניהול זיכרון) והשפעתם על ביצועי המחשב (מחזור ההבאה-והביצוע)

נושאים עיקריים

- ההשפעה של ההתפתחות הטכנולוגית על מבנה מעבדים מתקדמים
- מאפיינים של מעבדים ממשפחת אינטל 86 והשפעתם על ביצועי המחשב
- אוגרים במעבדים מתקדמים
- ארכיטקטורת "צינור הוראות" (pipelining)
- תור הוראות
- מצבי סיכון hazard
- שיטות לטיפול במצבי סיכון
- ביצוע במקביל – ארכיטקטורת SUPERSCALAR
- יחידה לחיזוי כתובת קפיצה (branch prediction)
- ארגון זיכרון
- אפני עבודה
- סוגי זיכרונות (זיכרון ראשי משני, זיכרון מטמון)
- סגמנטציה והרצה של כמה תכניות במקביל

- תרגום כתובת לוגית לכתובת פיזית זיכרון מדומה
- דפדפוף

הידע שירכשו התלמידים בפרק זה יכשיר אותם:

- להסביר את אופן מבנה המחשב המודרני
- להסביר מהו ארכיטקטורת צינור הוראות
- להסביר את המושג סיכון hazard
- להסביר בקטע תכנית נתונה אילו סיכונים קיימים
- להסביר כיצד ניתן להתמודד עם בעית סיכון
- לתאר את סוגי הזיכרונות
- להסביר את אופן השימוש בזיכרון מטמון
- להסביר יתרונות וחסרונות של שימוש בסגמנטציה ודפדוף

פעילויות לתלמיד

פעילות 13 – מעבדים מתקדמים

הרחבה

מקורות נוספים על אופן פעולת CPU וצינור הוראות (בעברית)
http://www2.eitan.ac.il/CpuBrain/map_site.asp

מקורות נוספים על אופן ארגון הזיכרון (באנגלית)
http://cne.gmu.edu/modules/vm/green/defn.html
http://www.cs.iitm.ernet.in/~sashi/csd/tutorial/VirtualStorage.html

מקורות נוספים על משפחת מעבדי אינטל (באנגלית)
http://www.ul.ie/~rinne/et4508.htm

מדינת ישראל
משרד החינוך
הגף לאישור ספרי לימוד
רחוב דבורה הנביאה 2, בנין לב-רם, ירושלים

29 ליוני 2003

שלום למורה;

המשרד יזם לאחרונה מהלך שמטרתו לקבל משוב על עזרי הוראה-למידה חדשים שנכללו בחוזר המנכ"ל "ספרי לימוד מאושרים". במסגרת זו אני פונה אליך באמצעות טופס זה בבקשה לקבל את חוות דעתך. חוות דעת תתבסס הן על עיון בספר והן על השימוש בו בכיתות. הערותיהם של המורים יועברו לגף אישור ספרי לימוד ירוכזו וייבדקו, כל הערה תיבדק לגופה. הערות שיימצאו ראויות להתייחסות יועברו מטעם גף אישור ספרי לימוד למו"ל. המו"ל יהיה מחוייב להתייחס אליהן במהדורה הבאה של הספר.
תודה לך, אם תעזוור/תעזרי לנו לשפר את הספר.

אנא מלא/י את השאלון ושלח/י אותו בדואר או בפקס עד לראשון בפברואר של כל שנה.

בתודה,

בנימין לוי

מנהל גף אישור ספרי לימוד

ההערכה של המורה על הספר

תאריך _____ שם הספר _____

שם המחבר _____

לכיתה_____ שם המורה_____ מספר טלפון_____

	קביעה	כן	לא	הערות או העמודים שבהם יש שיבושים
1.	הספר מכסה את כל הנדרש על פי תכנית הלימודים			
2.	הספר מדייק מבחינה מדעית			
3.	העברית בספר תיקנית			
4.	הספר מתאים מבחינה פדגוגית ודידקטית, לשכבת הגיל			
5.	הגרפיקה (מפות, איורים, צילומים, טבלאות וכו') משרתת את הבנת הטקסט			
6.	בספר אין אפליה בין המינים, אין תיאור סטראוטיפי של גברים ונשים			
7.	בספר אין פרסומת מסחרית, מפלגתית או פוליטית			
8.	בספר אין הוראות כתיבה			
9.	אין בספר פגיעה במיעוטים, בקבוצה, בעדה או בתרבות כלשהי			
10.	צורת הספר נאה והולמת את דרכי החינוך			
11.	הערות נוספות			

טלפונים : 5603534, 5303503, 5603004 פקס 5603488