



תכנית לימודים
מערכות מחשב ואסמבלר
בחלופה מטלת ביצוע, במדעי המחשב
(שלושים אחוז, לסמל בחינה 899283)
גרסה 23-6-2021

הזמן הנדרש

עיוני: 30 שעות

מעשי: 60 שעות

אוכלוסיית יעד

תלמידים שסיימו יסודות מדעי המחשב 1 ויסודות מדעי המחשב 2.

מבוא ורצינות

היחידה תטמיע הבנה עמוקה, מבוססת על הרצאות ועבודה עצמית מודרכת, של שלושה נושאי מפתח במדעי המחשב: (1) איך מערכות חומרה ותוכנה עובדות, (2) איך בונים מחשבים, (3) איך מתרגמים תוכניות (קומפילציה בסיסית). בוגרי היחידה יבינו "במו ידיהם" כיצד התוכניות שהם כותבים בשפות עיליות מתורגמות ומבוצעות על פלטפורמת החומרה המארכת, וכיצד ניתן לנצל את הפלטפורמה הזאת באופן מיטבי. בנוסף, התוכנית חושפת את התלמידים לסביבת עבודה שמשלבת רכיבי חומרה ותוכנה באופן הדוק, כפי שנהוג בפרויקטי embedded systems.

מתודולוגיה

היחידה מלמדת מבנה מחשבים ושפות סף על ידי בניית מחשב מודרני, מאָלף עד תו. את המסע הקונסטרוקטיבי הזה התלמידים מתחילים משערים לוגיים בסיסיים, ומסיימים לאחר שישה פרויקטי בנייה במערכת מחשב כללית שמעוצבת לבצע כל תוכנית שהיא. כל הידע התיאורטי והפרקטי הנדרש לבניית המחשב ניתן במסגרת השיעורים וחומרי הלימוד של היחידה. בנוסף לבניית המחשב, התלמידים יפתחו אסמבלר לתרגום תכניות סימבוליות לקוד בינארי. את האסמבלר ניתן לפתח בכל שפת תיכנות עילית כגון פייתון, C#, ג'אווה, C/C++, לפי הנחיות המורים בכל בית ספר.

התלמידים יבנו את כל הרכיבים (chips) המתוארים ביחידה בשפה פשוטה לאפיון חומרה (HDL), או Hardware Description Language) ויבדקו אותם על סימולטור חומרה, כנהוג בתעשייה. סימולטור החומרה הוא תוכנית מחשב חינוכית שרצה על כל מחשב אישי ומערכת הפעלה. אי לכך, התלמידים לא נדרשים לשום ציוד מיוחד פרט לגישה למחשב אישי, בבית הספר או בבית. בסיום היחידה (או בפרויקט המשך) ניתן לממש את המחשב שנבנה ב-HDL על כרטיס חומרה המבוסס על טכנולוגיית FPGA. המימוש הפיזי הוא פרויקט רשות טכני שאינו חלק מחובות היחידה.

כל חומרי הבנייה של המחשב (קבצי HDL שלדיים, תוכניות בדיקה, קבצי פלט להשוואה) יסופקו מראש. כמו כן, כל כלי התוכנה של הקורס (סימולטור חומרה, אמולטור CPU, אסמבלר) יסופקו מראש. חומרי הבנייה וכלי התוכנה ארוזים בקובץ זיפ חינמי שהתלמיד מוריד למחשב אישי או לדיסק נתיק בתחילת הקורס. מרגע זה ואילך, כל עבודת התלמיד נשמרת בתיקיית קורס ייעודית. תלמידים שאין להם גישה למחשב אישי פרטי יכולים לשמור את תיקיית הקורס בזיכרון נתיק, ולהריץ את כל כלי התוכנה הנדרשים להשלמת היחידה מתוך התיקייה הזאת.

הרצאות

היחידה מבוססת על סדרת הרצאות וסדרת פרויקטים. כדי להקל על הטמעת היחידה החדשה במערכת, ריכזנו את כל ההרצאות שעוסקות בבניית המחשב והאסמבלר במסמך הזה. ההרצאות נגישות בשני פורמטים: סרטוני יוטיוב, ומצגות פאוורפוינט. חומרים אלה ניתנים כהצעה בלבד. מורי היחידה מוזמנים לערוך את המצגות לפי שיקול דעתם, ולהכין את השיעורים הפרונטלים בעזרת הרצאות הוידאו, או בלעדיהן, לפי שיקול דעתם.

פרויקטים

היחידה כוללת שישה פרויקטי פיתוח מעשיים במסגרתם התלמידים בונים את כל הרכיבים שמתוארים בהרצאות. מבחינה פדגוגית, הפרויקטים הללו הם הנדבך החשוב ביותר ביחידה. התלמידים יכולים לבצע את הפרויקטים כעבודת בית, כתרגול מודרך במעבדת בית הספר, או כשילוב של השניים. לשיקול דעת המורה, התלמידים יכולים לפתח את הפרויקטים (או חלק מהם) כבודדים, או כזוגות. המסמך הנוכחי כולל תיאור קצר של כל פרויקט, וקישור להנחיות הפרויקט המפורטות.

בדיקה

כל רכיבי החומרה שהתלמידים יפתחו ממומשים כתוכניות HDL ששמורות בקבצי טקסט פשוטים. ניתן לבדוק כל תוכנית כזאת בשלוש דרכים:

1. התלמידים יבדקו את נכונות התוכנית ע"י הרצתה על סימולטור החומרה שמותקן על המחשב האישי.
2. התלמידים יגישו את התוכנית לשרת בדיקה אוטומטי שבדק את נכונות התוכנית באופן דומה.
3. המורים יבדקו את איכות הקוד שהוגש, לפי שיקול דעתם (יעילות, סגנון, קריאות, תיעוד).

מבנה היחידה

מבוא

חלק ראשון: חומרה (פרקים 1,2,3,5)

לאחר הקדמה של נושאי תאוריה רלבנטים, התלמידים יבנו קבוצה של שערים לוגיים בסיסיים, מחברים (adders), יחידה אריתמטית לוגית (ALU), יחידות זיכרון (RAM, ROM), מעבד (CPU), התקני קלט-פלט, ומחשב-על-צ'יפ (microprocessors). המחשב מבוסס על מתווה מכונת פון נוימן ובנוי להריץ תוכניות כתובות בשפת מכונה. המחשב מחובר למקלדת רגילה ולמסך bitmap מסומלץ ומאפשר כתיבה והרצה של תוכניות גרפיות ואינטראקטיביות כגון משחקי מחשב פשוטים.

חלק שני: תוכנה (פרקים 4,6)

התלמידים יכתבו תוכניות בשפת סף ויבצעו אותן על המחשב אותו הם בנו בחלקה הראשון של היחידה. תוכניות אלה נכתבות בשפת מכונה סימבולית ומתורגמות ע"י אסמבלר נתון. לאחר התנסות עם האסמבלר

הנתון, התלמידים יכתבו אסמבלר בעצמם, וזאת ע"י מימוש API נתון והרצת תוכניות בדיקה נתונות. פרויקט בניית האסמבלר משלב יכולות שהתלמידים פיתחו ביסודות מדעי המחשב 1+2 עם חומרי היחידה הנוכחית. בפרט, התלמידים יכתבו תוכנית מונחית עצמים שמבוססת על מספר מחלקות ומטמיעה טכניקות תיכנות שונות כגון עבודה עם קבצים, עיבוד מחרוזות, טוקניזציה, ניתוח סינטקס, וטבלאות עירבול.

חלק שלישי: מערכות מחשב מודרניות (פרק 7)

חלקה האחרון של היחידה הוא גמיש, ומבוסס על 3 שעות הרצאה במסגרתן מורים יכולים לבחור לחשוף את התלמידים למגוון נושאי הרחבה אפשריים. לדוגמא: ייצוג ועיבוד טיפוסי נתונים שונים (טקסט, גרפיקה, אודיו), pipelining, זיכרון מטמון, מעבדים גרפיים (GPU), אופטימיזציה, מימוש רכיבים פיזי (FPGA), ונושאים בקומפילציה.

מבוא

(1 שעה עיוני, 1 שעה מעשי)

הרצאת המבוא עוסקת בשלושה נושאים. ראשית, נציג לתלמידים את סיפור הדרך, שיסקור בעיקר את סידרת הפרויקטים שיש לבצע. שנית, נדון במספר נושאים כלליים בבניית מערכות: תיכנון מודולארי וחלוקה לרכיבים עצמאיים, הפשטה ומימוש, ובדיקה מדורגת (unit testing). שלישית, נסביר כיצד הבנה עמוקה של נושאי חומרה וקומפילציה בסיסית הופכת תוכניתנים "רגילים" למפתחי תוכנה מתוחכמים שיכולים להשתלב בפרויקטי פיתוח מורכבים. כדי לתת מוטיבציה וקונטקסט, ניתן להזכיר חברות שמפתחות מוצרים שמשלבים רכיבי חומרה ותוכנה כגון מובילאיי (נהיגה אוטומטית, נמכרה לאינטל), פריימסנס (ראייה ממוחשבת, נמכרה לאפל), מלאנוקס (תקשורת, נמכרה לאנבידיה), ודוגמאות נוספות מתחומי מיכשור רפואי, משחקי מחשב, מדיה דיגיטלית, ועוד.

נושאי ההרצאה בפרק המבוא:

[0.1 מנאנד לטטריס](#)

[0.2 תוכנית הקורס](#)

[0.3 הפשטה ומימוש](#)

[0.4 פרויקט 0 \(תיאור\)](#)

פרויקט 0: במסגרת מטלה זאת התלמידים יבצעו שלוש משימות פשוטות: התקנת חבילת התוכנה של היחידה, התקנת עורך טקסט, ותירגול הגשת קובץ zip. [קישור לפרויקט 0 \(הנחיות מפורטות\)](#)

פרק 1: לוגיקה בוליאנית

(4 שעות עיוני, 10 שעות מעשי)

בפרק זה התלמידים מקבלים הקדמה ללוגיקה ואלגברה בוליאנית, ולומדים כיצד לממש אופרטורים ופונקציות בוליאניות באמצעות שערים לוגיים. לאחר מכן נלמד כיצד לבנות שערים לוגיים ורכיבים (chips) באמצעות שפה לאיפיון חומרה (Hardware Description Language, או HDL), וכיצד לבדוק ולהדמות את פעולת הרכיבים באמצעות כלי תוכנה שנקרא "סימולטור חומרה".

מטרות ביצועיות:

1. התלמידים יסבירו מהי טבלת אמת, ויכתבו דוגמא של טבלת אמת של שער לוגי כלשהו
2. התלמידים יקבלו תוכנית HDL שמממשת שער לוגי של שני קלטים, ויכתבו את טבלת האמת של השער
3. התלמידים יקבלו הגדרת ממשק של שער לוגי פשוט כלשהו, ויממשו את השער ע"י כתיבת תוכנית HDL
4. התלמידים יקבלו דיאגרמת שערים (מימוש ויזואלי) של שער לוגי פשוט כלשהו, ויממשו את השער ע"י כתיבת תוכנית HDL
5. התלמידים יסבירו מדוע ניתן לממש כל פונקציה בוליאנית בעזרת שלושת השערים And, Or, Not
6. התלמידים יקבלו את טבלאות האמת של שערי And, Or, Nor, הגדרה של פונקציה בוליאנית כלשהי של שלושה משתנים שמשתמשת בשערים הללו, ושלושה ערכי אמת של שלושת המשתנים הללו. התלמידים יחשבו את ערך הפונקציה על הקלט הזה
7. התלמידים יקבלו את טבלת האמת של שער Mux, ממשק של שער כלשהו שכדי לממש אותו צריך להשתמש בשער Mux, ויכתבו את מימוש השער בשתי דרכים: דיאגרמת שערים, ותוכנית HDL

הרצאות פרק 1

1.1 מושגי יסוד	1.8 בדיקת חומרה ב'
1.2 אלגברה בוליאנית	1.9 בניית רכיבים (Chips): סיכום
1.3 פונקציות בוליאניות	1.10 רצפי סיביות
1.4 נאנד (Nand)	1.11 פרויקט 1: רכיבים
1.5 שערים לוגיים	1.12 פרויקט 1: הנחיות
1.6 שפה לאיפיון חומרה (HDL)	1.13 פרספקטיבה
1.7 בדיקת חומרה א'	

פרויקט 1: בפרויקט זה התלמידים מממשים ובודקים את כל הרכיבים (chips) שתוארו בהרצאות פרק 1. בפרט, התלמידים מממשים שערי Not, And, Or, Xor, מולטיפלקסרים שונים, והרחבות 16-ביט של השערים הללו. השערים הלוגיים הבסיסיים אותם בונים בפרויקט זה מהווים אבני בנייה למימוש שני רכיבים מרכזיים בכל מערכת מחשב: יחידה אריתמטית-לוגית (Arithmetic-Logic Unit או ALU) ויחידת זיכרון (RAM). את הרכיבים האלה התלמידים יכירו ויבנו בפרקים 2 ו-3, בהתאמה. [קישור לפרויקט 1 \(הנחיות מפורטות\)](#)

פרק 2: אריתמטיקה בוליאנית

(4 שעות עיוני, 9 שעות מעשי)

בפרק זה נלמד כיצד להשתמש בקודים בינאריים (רצפים של סיביות, או ביטים) כדי לייצג מספרים שלמים (חיוביים ושלייליים), וכיצד לעצב לוגיקת שערים שמשתמשת ברכיבים שבנינו בפרק הקודם כדי לבצע פעולות אריתמטיות על מספרים שלמים. גולת הכותרת של הפרק היא תיאור ובנייה של יחידה אריתמטית-לוגית, או ALU – רכיב חומרה שמבצע משפחה של פעולות אריתמטיות ולוגיות בסיסיות. ה-ALU שנבנה בפרק זה יהיה הרכיב המרכזי של המעבד, CPU, אותו נבנה בפרק 5 בהמשך התוכנית.

מטרות ביצועיות:

1. התלמידים יבצעו המרה של מספר שכתוב בשיטה הבינארית למספר שכתוב בשיטה העשרונית
2. התלמידים יבצעו המרה של מספר שכתוב בשיטה עשרונית למספר שכתוב בשיטה הבינארית
3. התלמידים יקבלו שני מספרים בינאריים, ויכתבו את סכומם כמספר בינארי
4. התלמידים יקבלו טבלה שמפרטת את הערכים הבינאריים של המספרים 8, ..., 1, 0, -1, ..., -7 לפי שיטת משלים ל-2, שני מספרים עשרוניים שלפחות אחד מהם הוא שלילי, ומשימה לחבר או לחסר את שני המספרים הללו. התלמידים יבצעו את הפעולה
5. התלמידים יקבלו הנחייה שהם פועלים על מחשב שרוחב המילה שלו הוא 4 סיביות, ויתנו דוגמא לשני ערכים בינאריים שהחיבור שלהם מייצר גלישה (overflow)
6. התלמידים יסבירו מהי יחידה אריתמטית לוגית, ALU
7. התלמידים יקבלו את טבלת האמת של ה-ALU, שני ערכים בינאריים של לכל היותר 4 סיביות, ופעולה אריתמטית או לוגית כלשהי. התלמידים יוכיחו באופן מילולי שה-ALU מבצע את הפעולה הנתונה באופן נכון
8. התלמידים יקבלו את המימשקים של הרכיבים HalfAdder ו-FullAdder, ויכתבו תוכנית HDL שממשת את הרכיב FullAdder
9. התלמידים יסבירו מדוע לא צריך לפתח חומרה מיוחדת לביצוע פעולות חיסור
10. התלמידים יקבלו את המימשקים של הרכיבים FullAdder ו-4Add, כאשר האחרון הוא רכיב שמחבר שני מספרים שרוחב כל אחד מהם הוא 4 סיביות. התלמידים יכתבו דיאגרמת רכיבים ותוכנית HDL שמממשים את המחבר 4Add באמצעות רכיבי FullAdder

2.1 מבוא	2.7, ALU, א'
2.2 ייצוג מספרים	2.8, ALU, ב'
2.3 מספרים בינארים	2.9 פרויקט 2, רכיבים
2.4 אריתמטיקה בינארית	2.10 רצפי סיביות
2.5 ייצוג מספרים שליליים, א'	2.11 פרויקט 2, הנחיות
2.6 ייצוג מספרים שליליים, ב'	2.12 פרספקטיבה

פרויקט 2: בפרויקט זה התלמידים יממשו ויבדקו את כל הרכיבים (chips) שתוארו בהרצאות פרק 2. בתחילה, התלמידים יבנו משפחה של מחברים (half adder, full adder, 16-bit adder) מהרכיבים אותם הם בנו בפרויקט 1. לאחר מכן, הם ישתמשו במחברים הללו ובמולטיפלקסרים שנבנו בפרויקט 1 כדי לבנות יחידה אריתמטית לוגית, ALU, שיודעת לבצע חישובים אריתמטיים ולוגיים שונים. [קישור לפרויקט 2 \(הנחיות מפורטות\)](#)

פרק 3: זיכרון

(4 שעות עיוני, 9 שעות מעשי)

בפרקים 1 ו-2 תיארו ובנינו רכיבים "קומבינטוריים" (combinational chips) שמגיבים באופן מיידי לכל שינוי בקלטם שלהם. בפרק הנוכחי נעצב רכיבים "סדרתיים" (sequential chips) שרגישים ומגיבים גם לקלטם שלהם, וגם להתקדמות השעון. בפרט, נלמד כיצד המחשב מייצג זמן, כיצד לבנות אוגרים – רכיבים שיודעים לאחסן מידע לאורך זמן – וכיצד לבנות יחידת זיכרון (RAM רצף של אוגרים שניתן לגשת לכל אחד מהם לפי כתובתו. כדי לבצע את המשימות הללו נציג רכיב זיכרון בסיסי בשם Data Flip-Flop שיודע "לשמור מצב" לאורך זמן, ונלמד כיצד לעצב לוגיקת שערים שמאפשרת גישה ישירה לכל אוגר בזיכרון, וקריאה וכתובה לתוך האוגר הנבחר.

מטרות ביצועיות:

1. התלמידים יסבירו כיצד המחשב מייצג את התקדמות הזמן
2. התלמידים יסבירו את הקשר בין אורך מחזור הזמן (cycle) של המחשב למהירות המחשב
3. התלמידים יסבירו באופן מילולי מדוע כשמקטינים את אורך מחזור הזמן (cycle) של המחשב יותר מדי, המחשב מתחיל לעשות טעויות חישוב
4. התלמידים יקבלו את המימשקים של הרכיבים Bit ו-4 Register, כאשר האחרון הוא אוגר של מספרים ברוחב 4 סיביות, ויכתבו תוכנית HDL שמממשת את הרכיב Register
5. התלמידים יקבלו את המימשקים של הרכיבים RAM4, Register, Mux, DMux, כאשר האחרון הוא זיכרון RAM של 4 אוגרים, ויכתבו תוכנית HDL שמממשת את הרכיב 4RAM
6. התלמידים יסבירו מהי פעולתו של שער Flip-Flop, וכיצד השער הזה בא לידי ביטוי במימוש של רכיבי זיכרון

7. התלמידים יקבלו את המימשקים של הרכיבים Bit, DFF, Mux, כאשר DFF הוא שער Flip-Flop ו-Bit הוא אוגר שמייצג סיבית בודדת, ויכתבו דיאגרמת רכיבים ותוכנית HDL שמממשת את הרכיב Bit
8. התלמידים יקבלו את המימשקים של הרכיבים 8RAM ו-64RAM שמייצגים, בהתאמה, יחידות זיכרון של 8 אוגרים ו-64 אוגרים, ויסבירו באופן מילולי כיצד ניתן לממש את הרכיב 64RAM בעזרת הרכיב 8RAM

הרצאות פרק 3

3.1 מבוא	3.8 אוגרים : מימוש
3.2 ייצוג זמן במחשב	3.9 זיכרון RAM : מימוש
3.3 שער	3.10 לוגיקה סדרתית
3.4 אוגרים	3.11 פרויקט 3 : רכיבים, א'
3.5 זיכרון RAM	3.12 פרויקט 3 : רכיבים, ב'
3.6 מונים	3.13 פרויקט 3 : הנחיות
3.7 שער Flip Flop	3.14 פרספקטיבה

פרויקט 3: בפרויקט זה התלמידים יממשו ויבדקו אוגרים ויחידות זיכרון מהשערים הלוגיים ומהרכיבים שהם בנו בפרק 1, ומשערי DFF. בפרט, התלמידים יבנו אוגר שמייצג סיבית בודדת, אוגר שמייצג רצף של 16 סיביות, ורכיבי זיכרון (RAM) שמייצגים רצפים של אוגרים כאלה בכל אורך נדרש, תוך מתן גישה ישירה לכל אוגר.

[קישור לפרויקט 3 \(הנחיות מפורטות\)](#)

פרק 4: שפות סף

(5 שעות עיוני, 6 שעות מעשי)

בפרק זה התלמידים ייחשפו לשפות מכונה ולאמנות התיכנות בשפות סף, ויופתעו לגלות שכל התוכניות שהם כותבים בשפות עיליות, כולל שפות מונחות עצמים, ניתנות למימוש באמצעות פקודות מכונה אלמנטריות ולוגיקת goto פשוטה. בפרט, נלמד את העקרונות הבסיסיים של שפות מכונה, ונתרגל תיכנות בשפת מכונה (השמה, ביטויים אריתמטיים, לולאות, מצביעים). כדי לעשות זאת נדגים תוכניות קצרות רבות בשפת האסמבלי של המחשב אותו אנו בונים, ונריץ אותן על כלי תוכנה ששמו CPU emulator. תוכנה זאת מדמה את פעולת המחשב שאת בנייתו נשלים בפרק הבא.

מטרות ביצועיות :

1. התלמידים יסבירו מה התפקיד של תוויות (labels) בתכניות בשפת מכונה
2. התלמידים יסבירו איך מייצגים משתנים בתכניות בשפת מכונה
3. התלמידים יסבירו מה היתרונות של כתיבת תוכניות מכונה בשפה סימבולית על כתיבה בשפה בינארית

4. התלמידים יקבלו תיעוד של הפקודות הרלבנטיות בשפת מכונה, ויכתבו רצפי פקודות שכל אחד מהם מממש פעולה בסיסית כגון: השמת קבוע לאוגר, השמת ערך אוגר לאוגר, השמת ערך ביטוי אריתמטי לאוגר, קפיצה מותנית לכתובת בזיכרון. בכל מקום שכתוב לעיל "אוגר" הכוונה או לאוגר שנמצא במעבד, או לאוגר בזיכרון, דבר שמצריך גם טיפול בכתובות.
5. התלמידים יסבירו מהו CPU emulator, ולמה הוא משמש
6. התלמידים יקבלו תיעוד מלא של שפת המכונה, ויכתבו תוכנית שמממשת קפיצה מותנית. לדוגמא: חישוב max
7. התלמידים יקבלו תיעוד מלא של שפת המכונה, ויכתבו תוכנית שמממשת חישוב איטרטיבי. לדוגמא: count
8. התלמידים יקבלו תיעוד מלא של שפת המכונה, ויכתבו תוכנית שמממשת עבודה עם מצביעים. לדוגמא: חיפוש ערך בזיכרון
9. התלמידים יסבירו מדוע רצוי לסיים כל תוכנית בשפת מכונה בלולאה אינסופית
10. התלמידים יסבירו מהי מפת הזיכרון (memory map) של המסך, וכיצד משתמשים בה כדי לצייר על המסך

הרצאות פרק 4

4.11 תוכניות לדוגמא : לולאות	4.1 הקדמה
4.12 תוכניות לדוגמא : מצביעים	4.2 מושגי יסוד : פקודות
4.13 תוכניות לדוגמא : מערכים	4.3 מושגי יסוד : בקרה
4.14 שפת Hack : סקירה	4.4 מחשב Hack
4.15 שפת Hack : הגדרה	4.5 שפת Hack
4.16 פלט (output)	4.6 CPU Emulator
4.17 קלט (input)	4.7 פקודות בקרה
4.18 פרויקט 4	4.8 כתובות סימבוליות : משתנים
4.19 פרספקטיבה	4.9 כתובות סימבוליות : תוויות
	4.10 כתיבת תוכניות

פרויקט 4: לפרויקט שתי מטרות: לתרגל כתיבת תוכניות בשפת מחשב סימבולית (אסמבלי), ולתת הצגה ראשונית של ארכיטקטורת מחשב Hack אותה נסיים לבנות בפרק הבא בקורס. במהלך הפרויקט התלמידים יכתבו שתי תוכניות בשפת מכונה: תוכנית אלגברית שמשתמשת בלולאה פשוטה כדי להכפיל שני מספרים, ותוכנית אינטראקטיבית שמנהלת דיאלוג פשוט עם המשתמש ומציירת על המסך, תוך שימוש במצביעים. [קישור לפרויקט 4 \(הנחיות מפורטות\)](#)

פרק 5: ארכיטקטורת מחשב

(5 שעות עיוני, 9 שעות מעשי)

בפרק זה נציג את מתווה פון נוימן שמאפיין את רוב ארכיטקטורות המחשב בעולם, את עיקרון התוכנית המאוחסנת בזיכרון (stored program concept), את מחזור ה-fetch-execute, ואת תעבורת הנתונים

והפקודות בתוך המחשב (buses). לאחר מכן נלמד כיצד לבנות יחידת עיבוד מרכזית (CPU, Central Processing Unit) שנועדה לבצע באופן יעיל את פקודות המכונה אותן למדנו בפרק 4. את המעבד נבנה בעזרת לוגיקת שערים ורכיבי ה-ALU והאוגרים אותם בנינו בפרקים 1, 2, 3. כמו כן נלמד כיצד לבנות את יחידת הזיכרון המרכזית של המחשב (RAM), וכיצד לנהל התקני קלט / פלט באמצעות מפות זיכרון (bitmap).

מטרות ביצועיות:

1. התלמידים יסבירו מהו עיקרון התוכנית השמורה בזיכרון (stored program concept)
2. התלמידים יציירו תרשים כללי של מתווה פון נוימן
3. התלמידים יסבירו מהו מחזור fetch-execute
4. התלמידים יסבירו את תפקידו של מונה הפקודות (program counter)
5. התלמידים יסבירו את תפקידו של אוגר הפקודה, ובאיזה ארכיטקטורות מחשב מוכרחים להשתמש בו
6. התלמידים יסבירו למה משמש זיכרון מטמון (cache memory), ומה תפקידו בהאצת מהירות הפעולה של המחשב
7. התלמידים יסבירו באופן כללי כיצד ניתן להשתמש בפקודות HDL כדי לפרש (parsing) פקודות שכתובות בשפת מכונה בינארית
8. התלמידים יסבירו באופן כללי כיצד ניתן להשתמש בפקודות HDL כדי לנתב מיקרו-פקודות שכתובות בשפת מכונה בינארית לרכיבים שונים במעבד
9. התלמידים יסבירו כיצד המעבד מחליט אם התנאי שמנוסח בפקודת מכונה בינארית מסוימת אכן מתקיים
10. התלמידים יקבלו את הממשקים של הרכיבים CPU, RAM, ROM, ויכתבו תוכנית HDL שמאחדת אותם לרכיב-על שמתפקד כמחשב
11. התלמידים יסבירו איך מממשים את שגרת האתחול של המחשב (booting)
12. התלמידים יסבירו איך מדליקים ומכבים פיקסל בשורה נתונה ובעמודה נתונה במסך שחור לבן
13. התלמידים יסבירו איך יודעים על איזה מקש במקלדת המשתמש לוחץ

הרצאות פרק 5

5.1 מבוא	5.8 מעבד (CPU), מימוש, ב'
5.2 ארכיטקטורות מחשב	5.9 קלט / פלט
5.3 תהליך fetch-execute	5.10 זיכרון
5.4 מעבד (CPU), א'	5.11 מחשב
5.5 מעבד (CPU), ב'	5.12 פרויקט 5 : רכיבים
5.6 מעבד (CPU), ג'	5.13 פרויקט 5 : הנחיות
5.7 מעבד (CPU), מימוש, א'	5.14 פרספקטיבה

פרויקט 5: בפרויקט 5 התלמידים יבנו ויאחדו את כל הרכיבים שתוארו בהרצאות הפרק (מעבד, זיכרון, קלט ופלט (לארכיטקטורת מחשב רב-שימושי שבנוי לבצע תוכניות שכתובות בשפת המכונה אותה הצגנו בפרק הקודם.

[קישור לפרויקט 5 \(הנחיות מפורטות\)](#)

פרק 6: תרגום תכניות

(4 שעות עיוני, 16 שעות מעשי)

בפרק זה נתאר אתגרי ושיטות תרגום, ונלמד כיצד לפתח אסמבלר. לאחר דיון כללי בנושאים אלה, נציג את הפרויקט האחרון ביחידה: פיתוח אלגוריתם ותוכנית שמתרגמת תוכניות מהשפה הסימבולית שבה התלמידים תיכנתו בפרק 4 לשפה הבינארית של המחשב אותו התלמידים בנו בפרק 5. ההבנה של אתגרי ושיטות תרגום תקנה לתלמידים יתרונות פיתוח משמעותיים. בפרט, נחשוף את התלמידים לשורה של טכניקות תיכנות קלאסיות: ניתוח סינטקס, אלגוריתמי parsing, עיבוד מחרוזות, קריאה וכתיבה של קבצים, טוקניזציה, ייצור קוד, וניהול סמלים ע"י מימוש של טבלאות עירבול.

מטרות ביצועיות:

1. התלמידים יקבלו את תיעוד שפת המכונה (סימבולי ובינארי), ויתרגמו מספר פקודות סימבוליות מייצגות לקוד בינארי
2. התלמידים יקבלו את תיעוד שפת המכונה (סימבולי ובינארי), ויתרגמו תוכנית קטנה שכתובה בשפה הסימבולית לקוד בינארי
3. התלמידים יסבירו כיצד השימוש בכתובות סימבוליות מאפשר לטעון תוכנית (לאחר תרגומה) לאזורים שונים בזיכרון, בקלות רבה (relocatable code)
4. התלמידים יסבירו מהי טבלת סמלים (symbol table), ומהו תפקידה בתהליך התרגום
5. התלמידים יתארו את שני השלבים של אלגוריתם התרגום (two pass assembly)
6. התלמידים יסבירו מדוע כשמפתחים אסמבלר רצוי להתחיל בפיתוח אסמבלר בסיסי שפועל רק על תוכניות נטולות סימבולים
7. התלמידים יסבירו מדוע ניתן לממש אסמבלר בכל שפת תיכנות עילית

הרצאות פרק 6

[6.6 בניית אסמבלר, ב'](#)

[6.1 מבוא](#)

[6.7 פרויקט 6](#)

[6.2 תרגום פקודות](#)

[6.8 סקירה היסטורית](#)

[6.3 תרגום תוכניות](#)

[6.9 פרספקטיבה](#)

[6.4 תרגום סימבולים](#)

[6.5 בניית אסמבלר, א'](#)

פרויקט 6: המשימה היא לפתח תוכנית שמתרגמת תוכניות סימבוליות לקוד בינארי. כדי לבצע זאת, התלמידים יכתבו תוכנית משמעותית שתעשיר ותקדם את יכולות הפיתוח שלהם. האסמבלר הוא תוכנית מונחית עצמים מורכבת שמבוססת על מספר מחלקות ומטמיעה כמה טכניקות תיכנות קלאסיות. כדי לייעל ולפשט את הפיתוח, נספק לתלמידים API מפורט שמתעד את המחלקות הדרושות ומתאר את כל הפונקציות

הדרושות למימוש האסמבלר. כמו כן נספק תוכניות בדיקה שיאפשרו לפתח את האסמבלר באופן מדורג ומודולארי. פרויקט פיתוח האסמבלר מהווה סיום הולם ליחידת הלימוד הנוכחית, שכן הוא מאפשר לתלמידים להשתמש ביכולות שהם פיתחו ביסודות מדעי המחשב 1+2 כדי לבנות נדבך חשוב ומשמעותי בכל מערכת מחשב.

[קישור לפרויקט 6 \(הנחיות מפורטות\)](#)

פרק 7: מערכות מחשב מודרניות

(3 שעות עיוני)

ששת הפרקים הראשונים של היחידה מאופיינים ע"י עבודה עצמית אינטנסיבית: כל רכיבי החומרה והתוכנה שנלמדו בששת הפרקים ממומשים ע"י התלמידים. הפרק השביעי והאחרון מספק הזדמנות לסקור בקצרה מספר נושאים במערכות מחשב ואסמבלר שלא נדונו עד כה ביחידה, או נדונו בקיצור נמרץ. כדי לאפשר גמישות, פרק 7 מציע אסופה של נושאי בחירה, שנועדה לאפשר למורים לסיים את היחידה במגוון הרחבות אפשריות. כללית, אנו מציעים שלושה כיווני הרחבה אפשריים:

אופטימיזציה: הרחבות חומרה ותוכנה אפשריות שמטרתן לייעל את פעולות המחשב. ברמת החומרה, נדון בנושאים כגון pipelining, זיכרון מטמון, מיקבול, ומאיצים גרפיים. ברמת התוכנה, נציג אלגוריתמים אלגבריים יעילים למימוש פעולות אריתמטיות וגרפיות במרחב הסיביות.

מימוש פיזי: במהלך היחידה התלמידים פיתחו רכיבים (chips) רבים ע"י כתיבת תוכניות HDL שנבדקו והורצו על סימולטור חומרה, בדיוק כמו שמהנדסי חומרה עובדים בתעשייה. כעת נסקור כיצד ניתן לממש את הרכיבים הללו בסיליקון, תוך שימוש בטכנולוגיות ASIC ו-FPGA.

קומפילציה: כיצד ניתן לאפשר למחשב שבנינו במהלך היחידה לבצע תוכניות שכתובות בשפות עיליות? כדי להתמודד עם האתגר הזה, נסקור בפני התלמידים בקצרה כמה נושאים רלוונטים כגון מכונות וירטואליות, שיטות לניהול ומיחזור זיכרון, וטכניקות קומפילציה בסיסיות.