

פרויקט גמר בקורס קומפילציה סמסטר חורף 2026

תוכן עניינים

1.	אופן ההגשה	3
2.	פרויקט המהדר	3
3.	תיאור פעולת המהדר	3
3.1.	מה עושה המהדר?	3
3.2.	מפסק	3
3.3.	הממשק	4
3.4.	טיפול בשגיאות	4
4.	מימוש המהדר – כלים, שיטות ומבני נתונים	5
4.1.	שימוש בכלים flex ו-bison	5
4.2.	שיקולי מימוש	5
4.3.	סגנון תכנות	5
5.	כיצד להגיש את הפרויקט	6
5.1.	תיעוד	6
5.2.	מה להגיש	6
5.3.	בדיקת התכנית לפני ההגשה	6
6.	כיצד ייבדק הפרוייקט	7
6.1.	תהליך הבדיקה	7
6.2.	מבנה הציון	7
7.	שפת המקור – שפת התכנות CPL	8
7.1.	מבנה לקסיקלי	8
7.2.	מבנה תחבירי	8
7.3.	סמנטיקה	10
7.4.	תוכניות לדוגמה	10
8.	שימוש בתוכנת bison	11
9.	הסבר לגבי שימוש בכליב בינה מלאכותית AI	11
10.	דוגמת הרצה	12
10.1.	קלט	12
10.2.	דוגמת הרצה: פלט	12

1. אופן ההגשה

- יש להגיש בזוגות
- תאריך הגשה: **10.4.2026**
- יש להעלות את הפרויקט לאתר הקורס תחת "מטלה – פרויקט"
- בעמוד הראשון של ההגשה (readme.doc) יש לכלול:
 - שם מלא, מספר תעודת זהות, מספר טלפון וכתובות דוא"ל של שני הסטודנטים המגישים,
 - קישור למקום אחסון חיצוני שבו הועלו כל תכני הפרויקט (קישור ציבורי). דוגמאות למקומות אחסון GitHub, Google Drive וכדומה.
 - קישור לסרטון קצר ב YouTube-שבו מוצגים השלבים הבאים:
 - הרצת קובצי Bison וFLex
 - קומפילציה של כל קובצי ה-C-שנוצרו לקובץ cpm.exe
 - הרצת הקובץ cpm.exe על הקבצים min.cpl ו example.cpl - (המופיעים בהמשך חוברת זו)
- יש להגיש את כל הקודים (יש בעיה לעלות קובץ exe למודל)
- יש להגיש קובץ הסברים (כולל שימוש בכלי AI)
- את כל הקבצים יש לכלול בתוך קובץ zip.

2. פרויקט המהדר

בפרויקט זה עליכם לתכנן ולממש חלק קדמי וחלק מן החלק האחורי של מהדר, המתרגם תוכניות משפת המקור בשם CPL לשפת RISC-V או MIPS. שפת המקור CPL (Compiler Project Language) היא שפה דמוית פסקל או/וגם C, אך מוגבלת מהן בהרבה.

3. תיאור פעולת המהדר

3.1. מה עושה המהדר?

המהדר יבצע את כל שלבי ההידור (החלק הקדמי) כפי שנלמדו בקורס, החל בניתוח לקסיקלי, דרך ניתוח תחבירי ובדיקות סמנטיות, ועד לייצור קוד ביניים בשפת MIPS או V-RISC. ההידור יכלול טיפול בשגיאות, כפי שיפורט בהמשך.

המהדר יקבל קובץ קלט המכיל תוכנית בשפת CPL. כפלט, ייצר המהדר קובץ המכיל תוכנית בשפת MIPS או V-RISC. בנוסף, ייצר המהדר קובץ "רישום" (listing), שבו יפורטו שגיאות שהתגלו במהלך ההידור. (תוכלו להריץ בפועל את תוכניות ה-MIPS הנוצרות, בעזרת סימולטור של MIPS בשם MARS שניתן להוריד מ-<http://courses.missouristate.edu/KenVollmar/MARS/index.htm>). RISC-V משתמש בסימולטור בשם RARS.

3.2. מפקס

בפרויקט זה עליכם לממש מפקס בשיטת הניתוח העולה, כלומר באחת משיטות LR שנלמדו בספר: SLR, LALR, או LR קנוני (LR(1)).

תוכלו לבחור באחת משתי אפשרויות לבניית המפקס:

1. שימוש בתוכנת bison.
2. בניית מנתח תחבירי לפי אחת מהשיטות המפורטות לעיל, ללא שימוש בכלים אוטומטיים (לא מומלץ).

אם תבחרו לכתוב בעצמכם מנתח תחבירי, ללא שימוש ב-bison, יהיה עליכם לבנות את טבלת המפסק. כיון שמדובר בדקדוק גדול למדי, קשה לבנות את הטבלה באופן ידני. ניתן להיעזר ב-bison לצורך בניית טבלת LALR (בלי להיעזר בו לבניית המפסק כולו).

3.3. הממשק

המהדר יהיה תוכנית המופעלת משורת הפקודה (Command Prompt / DOS). שמו של המהדר הוא cpm (קיצור של CPL to MIPS/RISC-V). קובץ הריצה צריך להיקרא cpm.exe. הקובץ עם הפונקציה הראשית של המהדר (main) צריך להיקרא cpm.c. קלט – המהדר מקבל כפרמטר יחיד שם של קובץ קלט (קובץ טקסט המכיל תוכנית בשפת CPL). הסיומת של שם קובץ הקלט חייבת להיות cpl או CPL. שורת הפקודה היא: cpm <file_name>.cpl

פלט – המהדר יוצר שני קבצי טקסט עם שם זהה לשם קובץ הקלט ועם סיומות כדלקמן: קובץ "רישום" (listing), עם סיומת lst או LST; אל הקובץ הזה מעתיק המהדר את התוכנית (ללא ההערות), ומדפיס אותה כאשר בתחילת כל שורה רשום מספר השורה. בקובץ זה מופיעות גם הערות השגיאה (אם ישנן). אם תוכנית הקלט תקינה – קובץ MIPS או V-RISC, עם סיומת s או S (למרות שהסיומת אינה חשובה :MIPS).

קובץ זה מכיל את תוכנית בשפת היעד שנוצרה. אם תוכנית הקלט מכילה שגיאה כלשהי (לקסיקלית, תחבירית או סמנטית) אין לייצר קובץ V-RISC או MIPS, גם לא קובץ ריק. דוגמה:

עבור קובץ קלט תקין בשם min.cpl ושורת פקודה cpm min.cpl ייווצרו שני קבצים min.lst וגם min.s

טיפול בשגיאות ממשק – במקרה של שגיאה בפרמטר הקלט, בפתיחת קבצים וכדומה, יש לסיים את הביצוע בצירוף הודעת שגיאה מתאימה למסך (stderr). במקרה כזה אין לייצר קובצי פלט. כחלק מהטיפול בשגיאות ממשק, יש לוודא שהסיומת של קובץ הקלט היא נכונה.

שורת חותמת – יש לכתוב שורת "חותמת" עם שם הסטודנט, אשר תופיע במקומות הבאים: stderr-ב

בקובץ ה-LST (בתחילתו)

בקובץ ה-MIPS או V-RISC (יש לכתוב כהערה של MIPS או V-RISC)

שימו לב: יש להקפיד היטב על כל הוראות הממשק

חשוב שממשק המהדר שתכתבו יהיה בדיוק כפי שמוגדר במטלה. ייתכן שבדיקת הריצה של תוכניתכם תיעשה בצורה אוטומטית, ובמקרה כזה תוכנית בעלת שם שונה או ממשק שונה תיכשל. וודאו גם שניתן להריץ את תוכניתכם כאשר נמצאים במדריך אחר, ולא המדריך שבו נמצאת התוכנית עצמה. אין לכתוב מהדר המסתמך על קיומם של קבצים נוספים, כגון קובץ המכיל את טבלת הפיסוק. אם המהדר מייצר קובצי-עזר בזמן הריצה, יש לדאוג למחיקת הקבצים הללו בסיום הריצה.

3.4. טיפול בשגיאות

ייתכן שתוכנית הקלט תכיל שגיאות מסוגים שונים:

- שגיאות לקסיקליות
- שגיאות תחביריות
- שגיאות סמנטיות

שימו לב:

במקרה של קלט המכיל שגיאה (מכל סוג שהוא) אין לייצר קובץ V-RISC או MIPS, גם לא קובץ V-RISC או MIPS ריק. לאחר זיהוי של שגיאה לקסיקלית, תחבירית או סמנטית, יש להמשיך בהידור מהנקודה שאחרי השגיאה.

4. מימוש המהדר – כלים, שיטות ומבני נתונים

4.1. שימוש בכלים flex ו-bison

במטלה זו יש באפשרותכם לשלב את השימוש בכלים האוטומטיים flex ו-bison. flex הוא כלי אשר מייצר באופן אוטומטי מנתחים לקסיקליים, bison הוא כלי לייצור אוטומטי של מנתחים תחביריים. אין הכרח להשתמש בתוכנות אלו, וניתן גם לכתוב בעצמכם את המנתח הלקסיקלי והמנתח התחבירי, ללא שימוש בכלים אוטומטיים.

4.2. שיקולי מימוש

כתיבת המהדר נועדה להיות תהליך לימודי, ותכנון מבני הנתונים והאלגוריתמים שלו צריך להיגזר מכך. אין לקבל החלטות מימוש עיקריות על סמך הנסיבות שבהן המהדר יורץ בפועל על ידי בודק המטלה. לדוגמה, נדרש שמימוש טבלת הסמלים יהיה מתוחכם יותר מאשר חיפוש לינארי ברשימה, למרות שכאשר יש מספר קטן של מזהים, זהו פתרון סביר.

ברוח זו נוסיף: במימוש המבנים שגודלם תלוי בקלט יש להעדיף הקצאת זיכרון דינמית על-פני הקצאה סטטית שגודלה חסום ונקבע מראש. לעומת זאת, במימוש המבנים שגודלם קבוע וידוע מראש עדיפה כמובן הקצאה סטטית. במבנים אלה יש גם להעדיף מימוש "מונחה טבלה", שבו מאוחסן המידע ב"טבלה" נפרדת, והקוד משמש לגישה לטבלה ולקריאתה.

טבלת המפסק (במקרה של מימוש מפסק ללא bison) – לא הכרחי לממש דחיסה של הטבלה. מחסנית המפסק (במקרה של מימוש ללא bison) – אסור לממש את המחסנית בצורה שעלולה להגביל את גודלה. טבלת הסמלים – אסור לממש את הטבלה בצורה שעלולה להגביל את גודלה. בנוסף, יש לדאוג לכך שחיפוש והוספה בטבלה יהיו מהירים. המנתח הלקסיקלי – ייקרא על ידי המנתח התחבירי, ויחזיר בכל פעם אסימון אחד בלבד. חוצץ הקלט – מותר להשתמש בחוצץ (buffer), שלתוכו ייקרא קובץ הקלט כולו. אם אתם מניחים חסם על גודלו של קובץ הקלט, חשוב מאוד לרשום הנחה זו בתיעוד, וכן בקובץ readme שאותו תגישו. כדאי שהגודל יהיה לפחות 10K. בדיקות סמנטיות ויצירת קוד – יש לבצע על ידי מימוש של הגדרה מונחית-תחביר מתאימה.

4.3. סגנון תכנות

התוכנית שתכתבו צריכה לעמוד בכל הקריטריונים הידועים של תוכנית כתובה היטב: קריאות, מודולריות, תיעוד וכו'.

5. כיצד להגיש את הפרויקט

5.1. תיעוד

יש לכתוב תיעוד בגוף התוכנית, כמקובל. תיעוד זה נועד להקל על קוראי התוכנית.

בנוסף, יש לכתוב **תיעוד נלווה**: מסמך נפרד, שאותו ניתן לקרוא באופן עצמאי, ללא קריאת התוכנית עצמה. ניתן לכתוב את התיעוד הנלווה בעברית או באנגלית.

לתיעוד הנלווה שתי מטרות עיקריות: הסברים על שיקולי המימוש, ותיאור מבנה הקוד.

התיעוד אינו מיועד למשתמש נאיבי של המהדר, אלא לבודק הפרויקט, המכיר היטב (יש לקוות) את נושאי הקורס.

אין לחזור על דברים מובנים מאליהם, כגון שיש פעולות shift ו-reduce, אלא להבהיר את ההתלבטויות בין פתרונות שונים, ולהצדיק את ההחלטות שנעשו. בין השאר, יש לדון בנקודות אלה:

- מבנה הנתונים שנבחר לשמש כטבלת סמלים.
- מתי מעודכנת טבלת הסמלים, והאם מלים שמורות מוכנסות לטבלה זו.

לגבי הקוד, יש לתאר את החלוקה למודולים, תלויות הדדיות בין מודולים, מבנה זרימת הבקרה, פונקציות ראשיות, וכד'.

5.2. מה להגיש

1. הדפסה של התוכנית – (קובצי המקור – ראו הסבר בהמשך).
חשוב לארגן את ההדפסה בצורה שתקל על הקורא. למשל – ליצור הפרדה ברורה בין הקבצים השונים, ולסמן את שמות הקבצים.
2. הגשה ל MOODLE – יש לכלול את הקבצים הבאים (אין ליצור ספריות):
 - Readme.doc : שמות המבצעים, ת"ז, ד"א, קובץ טקסט פשוט, שיכלול את המידע הבא:
 - החלטות מימוש
 - קובצי המקור של התוכנית שכתבתם, כלומר כל קבצי ה-c וה-h, lex ו-y. קובץ הרצה cpm.exe.
 - אין צורך להגיש הדפסה של קובצי הפלט שנוצרים מ-flex או bison.
3. קובץ cp_pr_NAME1_NAME2.zip, כאשר Name1 מציין את השם המלא (שם פרטי ושם משפחה) של סטודנט 1 ו Name2 מציין את השם המלא של סטודנט 2.
שם הקובץ ייכתב באנגלית בלבד!
קובץ זה יכלול את כל החומר יש לשים בתיקיה מטלה פרויקט באתר הקורס. אם לא ניתן לעלות לאתר קבצי zip אז יש לשנות את הסימנים של קובץ txt. מספיק שרק אחד מבני הזוג יגיש את הפרויקט.

5.3. בדיקת התכנית לפני ההגשה

לקלטים תקינים, ייווצר קובץ V-RISC/MIPS המכיל תוכנית בשפת RISC-V או MIPS. השתמשו במפרש של שפת V-RISC או MIPS, שנקרא RARS או MARS. בעזרתו תוכלו להריץ את תוכנית ה-MIPS שיצרתם וכך לבדוק את תקינות הקוד המיוצר.

6. כיצד ייבדק הפרוייקט

6.1. תהליך הבדיקה

הבדיקה תכלול הרצה של המהדר שלכם על קלטים רבים, קריאת התיעוד הנלווה, וקריאה חלקית של קובצי המקור. בדיקות הרצה יכללו, בין השאר:

- הרצה על קלט תקין ובדיקת הפלט (באמצעות המפרש של MARS, וגם בדיקה יבשה).
 - בדיקת התגובות לשגיאות ממשק (פרמטר שגוי, סיומת השם שגויה, וכו').
 - בדיקת ההתמודדות עם שגיאות לקסיקליות, סמנטיות ותחביריות.
 - איכות מימוש טבלת הסמלים והמחסנית
- לדוגמה, בדיקת יכולת הטיפול של המהדר בתכנית קלט שיש בה מספר גדול - 250 לפחות - של מזהים.

6.2. מבנה הציון

ביצועי המהדר על קלטים תקינים.	60-55%
טיפול בשגיאות מכל הסוגים.	15-20%
החלטות מימוש, הסבר לגבי שימוש בכלי AI, בחירת מבני נתונים, מודולריות, כתיבת קוד כנדרש.	15%
תיעוד והגשה בהתאם לנדרש (שמות הקבצים, לינקים והפניות וכו').	10%

7. שפת המקור – שפת התכנות CPL (Compiler Project Language)

7.1. מבנה לקסיקלי

בשפה CPL ישנם אסימונים הבאים:

Reserved words

Reserved words

```
break  case  const      default do  else  end   for   if    int
main   print   real     read   start  string switch till
then   var     while    when
```

Reserved symbols

```
( ) { }
, : ; !
```

Composed tokens

```
id:          letter (letter|digit)*
num:         digit+ | digit+.digit*
relop:       == | != | < | > | >= | <=
addop:       + | -
mulop:       * | /
assignop:    :=
orop:        ||
andop:       &&
sentence:    "(letter|. , | ! | ? | | digit)* "
```

Where: (Note: digit and letter are not tokens)

```
digit: 0 | 1 | ... | 9
letter: a | b | ... | z | A | B | ... | Z
```

הבהרות:

1. בין האסימונים יכולים להופיע תווי רווח (space), תווי טאב (\t), או תוויים המסמנים שורה חדשה (\n).
תוויים כאלה חייבים להופיע כאשר הם נחוצים לצורך הפרדה בין אסימונים (למשל, בין מלה שמורה לבין מזהה). בשאר המקרים, האסימונים יכולים להיות צמודים זה לזה, ללא רווח.
2. אורכו של מזהה id מוגבל – עד 9 תוויים בלבד.
3. הערות בתוכנית מופיעות בין הגבולות /* */ (כמו בשפת C).
מותר להניח שבקלט שתקבלו אין הערה שגולשת מעבר לסוף שורה, ואין הערה שמכילה את הרצף "/*" (סימן של סוף הערה) בתוכה, לפני סופה.

7.2. מבנה תחבירי

CPLG - Grammar for the programming language CPL

PROGRAM → main id start DECLARATIONS STMTLIST end

DECLARATIONS → var DECLARLIST CDECL
| ε

DECLARLIST → DECLARLIST DECL
| DECL


```

DECL ->  TYPE : LIST

LIST ->  id , LIST
        |   id ;

TYPE ->  int
        | real
        | string

CDECL -> const TYPE id assignop num; CDECL
        | ε
        /* the value of id should not be changed during the program*/

STMTLIST → STMTLIST STMT
          | ε

STMT → ASSIGNMENT_STMT
      | id assignop sentence;
      | CONTROL_STMT
      | READ_STMT
      | WRITE_STMT
      | STMT_BLOCK

WRITE_STMT → print(EXPRESSION);
            | print(sentence);

READ_STMT → read(id);

ASSIGNMENT_STMT → id assignop EXPRESSION;

CONTROL_STMT → if (BOOLEXP) then STMT else STMT
              | while (BOOLEXP) STMT_BLOCK
              | for(ASSIGNMENT_STMT; BOOLEXP; STEP) STMT_BLOCK
              | do STMT_BLOCK till(BOOLEXP)
              /*do the block until BOOLEXP is true*/
              | SWITCH

STMT_BLOCK → { STMTLIST }

SWITCH → switch (CHOICE) { CASES }

CHOICE → id
        | num

CASES → case num: STMTLIST break; CASES
       | default: STMTLIST

STEP → id assignop id addop num
      | id assignop id mulop num

BOOLEXP → BOOLEXP orop BOOLTERM
         | BOOLTERM

BOOLTERM → BOOLTERM andop BOOLFACTOR
          | BOOLFACTOR

BOOLFACTOR → ! (BOOLFACTOR) /*Meaning not BOOLFACTOR*/
            |EXPRESSION relop EXPRESSION

EXPRESSION → EXPRESSION addop TERM
            | TERM
  
```

TERM → TERM mulop FACTOR
 | FACTOR

FACTOR → (EXPRESSION)
 | id
 | num

7.3. סמנטיקה

- א. כל משתנה מוצהר רק פעם אחת במהלך התוכנית.
- ב. קבועים מספריים שאין בהם נקודה עשרונית הם מטיפוס `int`. אחרת הם מטיפוס `real`.
- ג. הטיפוס של ביטויים נקבע על ידי הארגומנטים המופיעים בהם.
 1. כאשר בביטוי מופיע משתנה או קבוע מטיפוס `real`, הטיפוס של הביטוי כולו הוא `real`.
 2. בכל מקרה אחר טיפוס הביטוי כולו הוא `int`.
 3. חילוק בין שני שלמים נותן את המנה השלמה שלהם.
- ד. פעולת השמה היא חוקית כאשר שני אגפיה הם מאותו טיפוס או שהאגף השמאלי הוא `real`.
- ה. משמעות השפה וקדימות האופרטורים הם סטנדרטיים, כמו בשפת C.

7.4. תוכניות לדוגמה

```

main Min /* Finding minimum between two numbers */
start
var real : a,b;
    string: d;

    read(a);
    read(b);
    d="the result is ";
    print(d);
    if (a<b) then print(a);
        else    print(b);
end
    
```

```

main ForPrint
start
var int:a;
    string: d;
    d="Pls enter a number ";
    print(d);
    read(a);
    /* Printing numbers between 1 and a*/
    for(i=1;; i<=a; i=i+1)
        {print(i);}
end
    
```

8. שימוש בתוכנת bison

במטלה זו יש באפשרותכם לשלב את השימוש בתוכנת bison. זהו כלי עזר לבנייה אוטומטית של מנתחים תחביריים. תוכנת bison מקבלת כקלט קובץ עם הגדרת הדקדוק של השפה, ומייצרת תכנית בשפת C, שהיא מנתח תחבירי עבור השפה המוגדרת.

מימוש טבלת המפסק והמחסנית

(למי שאינו משתמש ב-bison).
מותר לממש את טבלת המפסק בעזרת מערך דו-ממדי פשוט. לא נדרשת דחיסה של הטבלה.
יש לממש את מחסנית המפסק בצורה שלא תגביל את גודלה, כלומר צריך להשתמש בהקצאת זיכרון דינמית.

סגנון תכנות

התוכנית שתכתבו צריכה לעמוד בכל הקריטריונים הידועים של תוכנית כתובה היטב: קריאות, מודולריות, תיעוד וכו'.

בדיקת התכנית לפני ההגשה

בדקו היטב את ריצת התוכנית שלכם על קלטים מגוונים. נסו תוכניות קלט עם שגיאות שונות, כולל מקרי קצה (קובץ ריק, קובץ עם הכרזות בלבד, קובץ עם שגיאות)

הגשה

במטלה זו יש להגיש קובצי המקור:

- תכנית שנכתבה ללא bison: יש להגיש את כל קובצי התוכנית (קובצי h ו-c).
- תכנית שנכתבה בעזרת bison: יש להגיש את קובץ הקלט של bison שכתבתם, את הקבצים ש-bison מייצר כפלט, וכל קובץ אחר (h או c) שהוא חלק מהתוכנית.
אין צורך להגיש הדפסה של קובץ הפלט שמייצר bison.

הידור

חשוב שבדק המטלה יוכל לבצע הידור לתכניתכם. תוכנית שלא תעבור הידור לא תיבדק.
רשמו בקובץ readme הוראות מדויקות להידור תכנתכם.

9. הסבר לגבי שימוש בכלי בניה מלאכותית AI

בפרויקט זה מותר ואף חובה להשתמש בכלי AI. יש לציין במפורש באילו כלי AI השתמשתם.
יש לתאר את אופן השימוש בכלי ה-AI בכל אחד משלבי הפרויקט, ולהתייחס לנקודות הבאות:
• בשלב הניתוח הלקסיקאלי - יש להסביר כיצד נעשה שימוש בכלי והאם הכלי סיפק פתרון מתאים, ומה היה נדרש לתקן או לשפר באופן עצמאי.
• בשלב המנתח התחבירי - יש לתאר את תרומת כלי לתכנון או למימוש הדקדוק והחוקים, והאם התקבלו קשיים או שגיאות במהלך העבודה.
• בשלב המנתח הסמנטי ותרגום הקוד ל-קוד ביניים - יש לפרט אילו בעיות עלו במהלך המימוש והתרגום, כיצד כלי סייע (או לא סייע), ואילו התאמות או פתרונות פותחו בסופו של דבר.

בנוסף, יש להתייחס בקצרה לאתגרים המרכזיים שהתגלו במהלך העבודה, ולתאר כיצד נפתרו בסופו של דבר – בין אם בעזרת כלי AI ובין אם באמצעות עבודה עצמאית.

10. דוגמת הרצה

10.1. קלט

קובץ CPL (התוכנית מכילה שגיאה):

```
main Example start
var          int:x;

read(x) ;
if  x >= 0)

                then print(x);
else            print(0-x);
end
```

10.2. דוגמת הרצה: פלט

קובץ LST:

```
1.      main Example start
2.      var          int:x;
3.
4.      read(x) ;
5.      if  x >= 0)
6.                then print(x);
7.      else            print(0-x);
8.      end
```

ERROR line 5, column 4: Expected '(', found identifier instead.

בהצלחה

