מכללת כפר סבא

הצעת פרויקט - אביב אש

סמל מוסד: 470112

מכללה: מכללת Ort design כפר סבא

שם הסטודנט: אביב אש

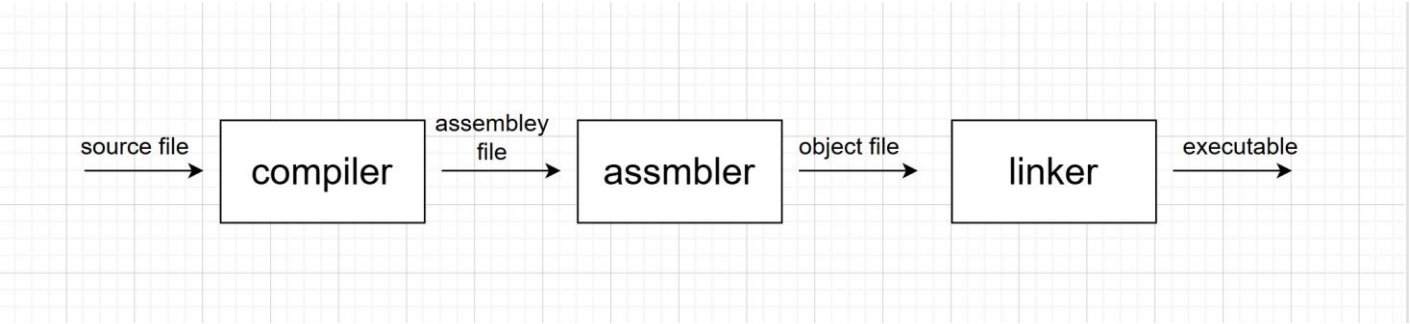
תז הסטודנט: 214887556

שם הפרויקט: tec – The Easy Compiler

רקע תאורטי בתחום הפרויקט:

קומפיילר/מהדר – הגדרה כללית:

בעולם המחשבים קומפיילר (או בשמו העברי: מהדר) הוא רכיב תוכנתי שתפקידו הוא להמיר בין שתי שפות תכנה, המהדר הקלאסי ימיר בין שפה עילית לשפת מכונה



ניקח לדוגמא את הקוד הבא הנכתב בשפה העילית "C" אשר נרצה להריץ:

```
1  int main()
2  {
3      int x = 10;
4      int y = 20;
5
6      x = x+y;
7
8      return 0;
9  }
```

שפות עליות כמו "C" ורבות אחרות עוצבו בצורה הדומה לשפה האנגלית המדוברת ושיהיה למתכנת נוח וקל להשתמש בהן אך המעבד לא בנוי בצורה שהוא יכול להבין בקלות את הוראות אלו. תפקידו של המעבד הוא להמיר את השפה העילית לשפת סף הבנויה בצורה קרובה לארכיטקטורת המעבד שאיתו אנחנו עובדים.

לצורך ההדגמה המהדר יצור לנו את קובץ הassembly הבא:

```
        .file   "dugma.c"
        .def    ___main;    .scl    2;  .type   32; .endef
        .text
        .globl  _main
        .def    _main;  .scl    2;  .type   32; .endef
_main:
LFB0:
        .cfi_startproc
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        andl    $-16, %esp
        subl    $16, %esp
        call    ___main
        movl    $10, 12(%esp)
        movl    $20, 8(%esp)
        movl    8(%esp), %eax
        addl    %eax, 12(%esp)
        movl    $0, %eax
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
LFE0:
        .ident  "GCC: (MinGW.org GCC-6.3.0-1) 6.3.0"
```

אז נשתמש בעוד מהדר הממיר קוד סף אל שפת מכונה

(דוגמאות למהדרים כאלו: NASM לארכיטקטורת x86_64 וTSEM לארכיטקטורת x86)

```
4C 01 05 00 00 00 00 00 7C 01 00 00 0F 00 00 00 00 00 04 01 2E 74 65 78 74 00 00 00 00 00 00 00 00 00 00 00
30 00 00 00 DC 00 00 00 68 01 00 00 00 00 00 00 01 00 00 00 20 00 30 60 2E 64 61 74 61 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 40 00 30 C0 2E 62 73 73 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 00 30 C0 2F 34 00 00
00 00 00 00 00 00 00 00 00 00 00 00 24 00 00 00 0C 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 40 00 30 40
2F 31 35 00 00 00 00 00 00 00 00 00 00 00 00 00 38 00 00 00 30 01 00 00 72 01 00 00 00 00 00 00 01 00 00 00
40 00 30 40 55 89 E5 83 E4 F0 83 EC 10 E8 00 00 00 00 C7 44 24 0C 0A 00 00 00 C7 44 24 08 14 00 00 00 8B 44
24 08 01 44 24 0C B8 00 00 00 00 C9 C3 90 90 90 47 43 43 3A 20 28 4D 69 6E 47 57 2E 6F 72 67 20 47 43 43 2D
36 2E 33 2E 30 2D 31 29 20 36 2E 33 2E 30 00 00 14 00 00 00 00 00 00 00 01 7A 52 00 01 7C 08 01 1B 0C 04 04
88 01 00 00 1C 00 00 00 1C 00 00 00 04 00 00 00 2D 00 00 00 00 41 0E 08 85 02 42 0D 05 69 C5 0C 04 04 00 00
0A 00 00 00 0E 00 00 00 14 00 20 00 00 00 04 00 00 00 14 00 2E 66 69 6C 65 00 00 00 00 00 00 00 FE FF 00 00
67 01 64 75 67 6D 61 2E 63 00 00 00 00 00 00 00 00 00 00 00 5F 6D 61 69 6E 00 00 00 00 00 00 00 01 00 20 00
02 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00 00 00 00 00 01 00 00 00
03 01 2D 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 2E 64 61 74 61 00 00 00 00 00 00 00 02 00 00 00
03 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 2E 62 73 73 00 00 00 00 00 00 00 00 03 00 00 00
03 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 19 00 00 00 00 00 00 00 04 00 00 00
03 01 23 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 24 00 00 00 00 00 00 00 05 00 00 00
03 01 38 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 5F 5F 5F 6D 61 69 6E 00 00 00 00 00 00 00 20 00
02 00 2E 00 00 00 2E 72 64 61 74 61 24 7A 7A 7A 00 2E 65 68 5F 66 72 61 6D 65 00 2E 72 64 61 74 61 24 7A 7A
7A 00 2E 65 68 5F 66 72 61 6D 65 00
```

סוגים שונים של מהדרים:

Source compiler – ממיר את הקוד הראשוני (בשפה העילית) היישר לשפת מכונה לדוגמא -רוב המעבדים של השפות C וCPP משתמשים בצורת הידור זאת

Intermediate Compiler - ממיר את הקוד הראשוני לשפת ביניים, אז שפת הביניים עוברת מהדר נוסף הממיר אותה לשפת מכונה. לדוגמא – המהדר של #C ממיר את קוד המקור לשפת CIL ומשם עובר עוד מהדר הממיר את הקוד לשפת מכונה

Transpiler – ממיר קוד משפה עילית אחת לשנייה לדוגמא – קוד בTypeScript יומר תחילה לJS בשלב הראשוני בהידורו (וזה נעשה בעזרת Transpiler)

סוגים שונים של שפות תכנות:

שפות תכנות פרוצדורליות - שפות כמו C ופסקל, בשפות אלו, התוכנית מורכבת מרשימת פקודות המבוצעות בסדר שלב אחרי שלב.

שפות תכנות מונחות עצמים -שפות כמו #C וJAVA. בשפות אלו, התוכנית בנויה ממודולים הנקראים אובייקטים, שכל אחד מהם כולל נתונים (תכונות) ופעולות (מתודולוגיות).

שפות תכנות פונקציונליות – שפות כמו Haskell וErlang. בשפות אלו, הפוקנציה היא יחידת התכנות העיקרית. התכנות נעשה דרך קריאה לפונקציות, ולא על ידי שינוי מצב של משתנים.

שלבי הקומפילציה:

תהליך הקומפילציה מתחלק לשני שלבים מרכזיים,

ה-**front end** וה-**beck end**.

Front end:

שלב זה מתעסק בניתוח הקוד והמרתו לתצורת ביניים (IR) שאיתה השלב השני (beck end) יכול לעבוד. הוא עוסק בניתוח לוגי ובהבנה של המבנה והמשמעות של הקוד, וגם הוא מתחלק לכמה חלקים.

ניתוח לקסיקלי (Lexical Analysis)

השלב הראשון בfront end עוסק בחילוק הקוד הנתון ל**טוקנים**, אשר כל טוקן הוא יחידה בעלת משמעות כמו מילים שמורות, משתנים, מספרים, אופרטורים וכדומה.

לדוגמה, פיסת הקוד int x = 7; תוכל להיות מתורגמת לטוקנים

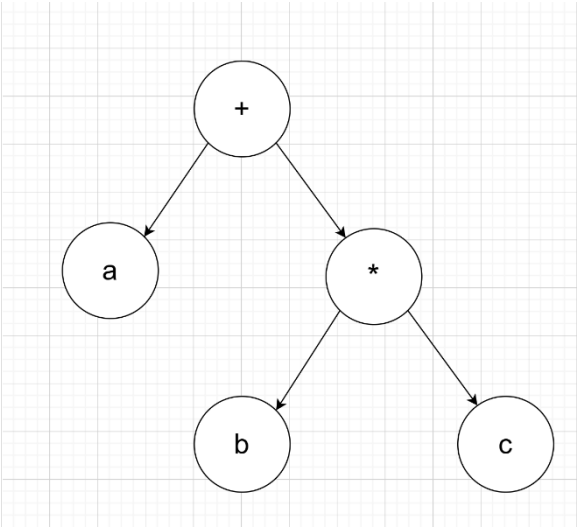
- Int (מילה שמורה)
- x (מזהה)
- = (אופרטור)
- 7 (מספר)
- ; (נקודה פסיק)
-

ניתוח תחבירי (Syntax Analysis)

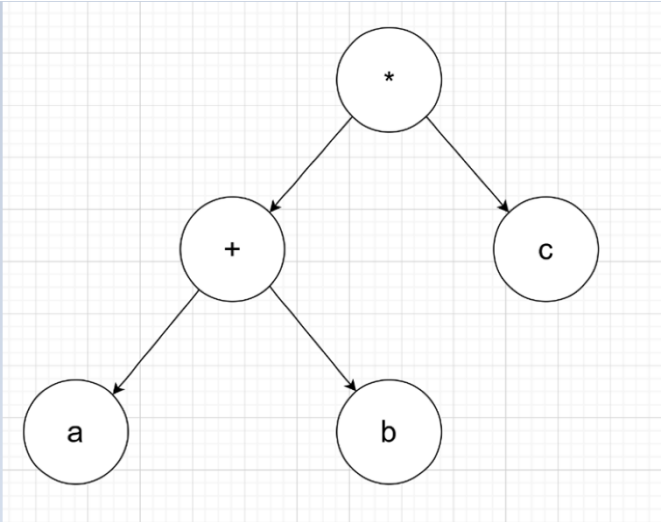
מטרה: השלב הבא הוא לבדוק אם רצף הטוקנים שנוצר בתהליך הקודם תואם לחוקי התחביר של השפה (הדקדוק שלה). השלב הזה יוצר את ה-עץ תחבירי או ה-עץ סמנטי (abstract syntax tree - AST), שמייצג את מבנה הקוד לפי כללי השפה.

דוגמה: אם הקוד הוא a + b * c, האנליזר התחבירי ייצור עץ תחבירי שמייצג את ההצהרה על המשתנה וההקצאה.

דוגמא לעץ סמנטי תקין:



דוגמא לעץ סמנטי לא תקין:



במהלך השלב הזה, אם יש בעיות תחביריות בקוד (כגון חוסר בסוגריים או סדר לא נכון של רכיבי השפה), יופיעו הודעות שגיאה.

ניתוח סמנטי (Semantic Analysis) – שלב הניתוח הסמנטי הוא שלב שמטרתו לעבור על ה**עץ הסמנטי** ולוודא שאין בקוד שגיאות סמנטיות כגון, בדיקת טיפוסים, וידוא התאמה של פרמטרים לפונקציות ועוד....

יצירת קוד הביניים – עבירה על העץ הסמנטי ותרגומו לקוד ביניים הישמש לשלבים הבאים ב-beck end

Back end:

שלב זה עוסק בניתוח קוד הביניים שהתקבל בשלב הfront end. המטרה בשלב זה היא לא "הבנה" או "ניתוח " של קוד המקור אלה לעבור על קוד הביניים ולהעביר אותו בצורה אופטימלית לשפת הסף

אופטימיזציה – שלב האופטימיזציה יעבור על קוד הביניים וייעל אותו בעזרת כמה שיטות, מחיקת שורות ללא משמעות, קיצור הקוד בדרכים שישאירו לו את אותה המשמעות וכו...

יצירת קוד הסף – לקיחת קוד הביניים שעבר אופטימיזציה ולהמיר אותו לקוד סף

טיפול בשגיאות:

בכל שלבי ההידור נרצה לתפוס שגיאות.

לדוגמא, כתיבת שם משתנה לא תקין (שגיאה תחבירית) או פנייה למשנה לא מוצהר (שגיאה סמנטית). טיפול בשגיאות הוא חלק קריטי בכל תהליך פיתוח תוכנה, ובפרט בקומפיילר. המטרה המרכזית היא להבטיח שהתוכנית שכתבת, או במקרה שלנו, הקוד שהקומפיילר מייצר, יתפקד בצורה נכונה, יציבה ובטוחה.נרצה להודיע למתכנת כשהקוד לא תקין ואיפה נמצאת השגיאה לנוחות ולפיתוח מהיר.

מכללת כפר סבא

תיאור הפרויקט

הפרויקט הינו בניית מהדר **source compiler** בשם Tec (The Easy Compiler) לשפה המומצאת שלי – **"Easy"**. המהדר יקבל קובץ הכתוב בשפה העילית ויתרגם אותו אל קובץ asm הכתוב בארכיטקטורת x86_64 המיועד לעבור הידור על ידי NASM אל שפת מכונה אשר נוכל להריץ.

תיאור השפה:

"Easy" הינה שפה הדומה לשפת C אך כוללת כמה שינויים והשראות משפות עיליות אחרות. **השפה כוללת:**

- חשיבות על סדר פעולות אריתמטיות
- טיפוסי משתנים סטטיים (int, short, char, bool)
- לולאות (for, while)
- מערכים (חד ממדיים)
- פונקציות
- תנאים (if, else)
- מצביעים וניהול זיכרון

השפה תטפל בשגיאות בזמן קומפילציה (שגיאות תחביריות, שגיאות סמנטיות, וכו...) ובשגיאות זמן ריצה (חלוקה באפס וכו...)

שפת easy תחשב לturing complete כלומר היא תוכל לעשות כל הפונקציונליות שמכונת טיורינג יכולה לעשות.

הבעיה האלגוריטמית

הבעיה האלגוריתמית טמונה במימוש ששת השלבי פיתוח הקומפיילר. כל שלב מציג בעיה אלגוריתמית שונה.

ניתוח לקסיקלי – חלוקת קוד המקור לטוקנים בעזרת **אוטומט דטרמיניסטי סופי**.

ניתוח תחבירי – שלב זה הוא מהעמוסים ביותר מבחינה אלגוריתמית בפיתוח המהדר, הוא כולל את יצירת העץ הסמנטי ועבודה אתו הכוללת מספר אלגוריתמים על עצים והכרעת מצבים בין בניית עצים שונים. שלב זה גם פועל על **אוטומט דטרמיניסטי סופי**.

ניתוח סמנטי – עבירה על העץ הסמנטי ובדיקה של **התקינות הסמנטית** שלו בעזרת **אוטומט דטרמיניסטי סופי**.

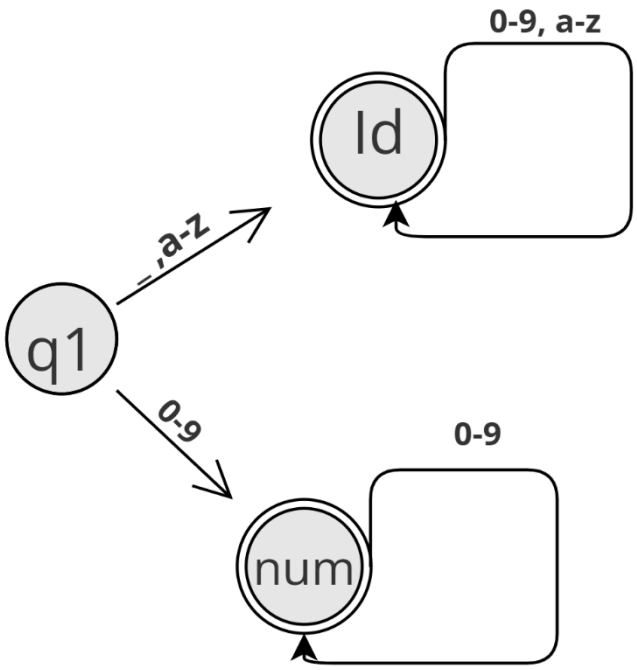
יצירת קוד הביניים – עבירה על העץ הסמנטי ותרגומו להצגת הביניים.

אופטימיזציה – עבירה על קוד הביניים ובדיקה על איזה מקודות ניתן לעשות לו אופטימיזציה בדרכים שונות.

יצירת קוד הסף – תרגום קוד הביניים אל קוד הסף.

אוטומט דטרמיניסטי סופי (DFA)

במהלך פיתוח המהדר נצטרך להשתמש **באוטומט דטרמיניסטי סופי** או **DFA** לדוגמא בשלב הניתוח הלקסיקלי נרצה להבדל אם רצף התווים שאנו סורקים הינו שם מזהה, מספר, או שהוא בכלל לא תקין. בשביל לממש זאת אולי נרצה לבנות את האוטומט הבא:



כפי שאנו רואים האוטומט למעלה יצליח לזהות מספרים ומזהים . כלומר: אם ניקח לדוגמה את המזהה sum נראה כי האוטומט יסיים בצומת Id וכך נדע שהוא מזהה. ציין כי עם קיבלנו קלט שהוא לא אחד מהקלטים האפשריים בצומת הנוכחית, נשלח לצומת מלכודת (Trap state) המציינת טוקן לא תקין

בשביל להעביר את האוטומט לקוד נרצה להשתמש ב**מטריצה דו ממדיית** או טבלה המחזיקה את כל המעברים בין שני צמתים

STATE	NUMBER	LETTER	_
q1	**num**	**Id**	**Id**
num	**num**	**TRAP**	**TRAP**
id	**Id**	**Id**	**TRAP**

תהליכים עיקריים בפרויקט

הבנת כל התהליכים השונים בבניית מהדר.

למידת assembly X86_64.

בניית המהדר בכל שלביו.

תיקון באגים ושיפור וקוד.

שפות תכנות

בפרויקט אשתמש בעיקר בשפת C++ בשילוב גבוהה של מונחות עצמים. בנוסף אשתמש בassembly x86_64 במקטעים.

סביבת עבודה

Visual Studio Code Version: 1.95.2

מכללת כפר סבא

לוח זמנים

	משימה	מועד אחרון להגשה/ביצוע
1	בחירת פרויקט הגשת נוסח ראשוני (First Draft) – הצעת ותיאור הפרויקט (PRD & Use Cases)	03/11/2024
2	הגשת נוסח סופי (final draft) - הצעת ותיאור הפרויקט (PRD & Use Cases) אישור ראשוני	13/11/2024
3	הגשה למשרד החינוך ואישור	01/12/2024
4	דו"ח	26/01/2025
5	מימוש lexer	1/2025
6	מימוש Parser	2/2025
7	הגשת דו"ח ביניים – התקדמות הפיתוח עד כה	23/02/2025
8	מימוש ניתוח סמנטי וקוד ביניים	3/2025
9	optimization and Code generation	4/2025
10	סיום פיתוח (Code Freeze)	אפריל 2025
11	בחינת מתכונת כולל תיק פרוייקט	אפריל 2025
12	סיום תיקון שגיאות (Bug Freeze) - סופי	אפריל 2025
13	הגשת תיק מלווה של הפרויקט סופי	אפריל 2025
14	הגנה על עבודת הגמר – פנימית - סופית	אפריל 2025
15	הגנה על עבודת הגמר - חיצונית	מאי 2025

מכללת כפר סבא

חתימות

חתימת הסטודנט:

X _____

חתימת רכז המגמה (ירון קובריגו):

X _____

חתימת מנחה הפרויקט (מיכאל צ'רנובילסקי):

X _____

אישור משרד החינוך:

X _____