פרויקט בקומפילציה – תיעוד חלק 3

מגישים:

אריאל יהודה 318296365

נדב אסטרוגנו 209270875

גיל סאיט 315816504

מימוש הקומפיילר:

מטרת הפרויקט היא לדמות Front-end של קומפיילר, כלומר לקחת שפת מקור --C ולהוציא ממנה ייצוג ביניים בקוד שלשות בשפת RISKI.

- עבור רוב הסימבולים בדקדוק המייצג את הקומפיילר בנינו מחלקה משלהם בעלת תכונות המתאימות להם, למשל עבור הסימבול EXP בנינו מחלקה בשם ExpSymbol ולה יש תכונות של Type, Place.

- כמו כן בנינו מחלקה ששמה Vec_buf ומייצגת buffer גדול של מחרוזות שהן קוד שלשות אותו מימשנו באמצעות vector של C++. במחלקה הנ"ל מימשנו פעולה בשם emit שמאפשרת לנו לבצע הוספות של קוד שלשות לבאפר כפי שלמדנו בהרצאה. כמו כן באמצעות העובדה שה-buffer ממומש באמצעות מערך (וקטור), יכולנו לממש את backpatch באמצעות האינדקסים בוקטור.

- כמו כן בנינו 2 טבלאות סמלים (הרחבה בהמשך) המנהלות רישום ומעקב אחר כל המשתנים והפונקציות המוגדרות בקוד. טבלאות הסמלים איפשרו לנו לבצע עבודה של בדיקות סמנטיות עבור scope של משתנים/פונקציות וכמו כן התאמה בין טיפוסים של משתנים/פונקציות.

- הקומפיילר שבנינו נשען על מימוש הפרסר שביצענו בחלק 2 של הפרויקט. כלומר, הפרסר מזהה חוקי גזירה מתאימים ובחלק 3 הוספנו קוד שמבצע בדיקות סמנטיות, עבודה מול טבלאות סמלים וemit ל-buffer במידת הצורך.

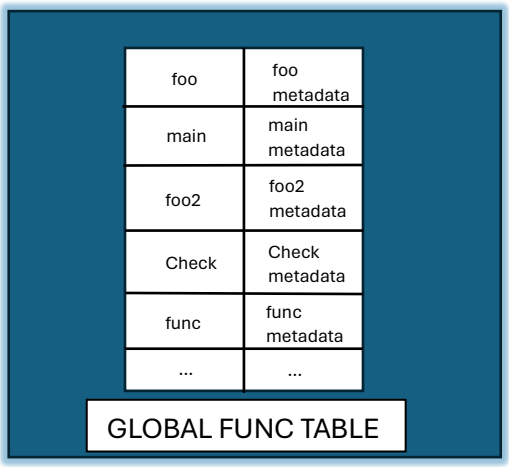
מבני הנתונים בשימוש:

יצרנו שתי טבלאות סמלים גלובליות: אחת לפונקציות ואחת למשתנים באופן הבא:

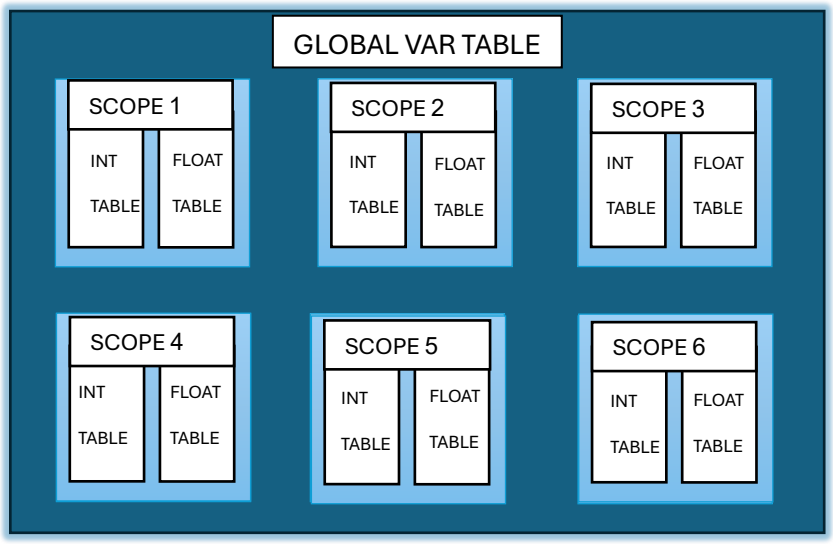
<u>טבלת סמלים לפונקציות:</u> באמצעות שימוש בmap של C++ מימשנו מפה של פונקציות שהוגדרו במהלך הקוד. שדה ה-key במפה הוא שם הפונקציות כאשר בשדה ה-value יש למעשה metadata על הפונקציה שמימשנו באמצעות מחלקה שיצרנו ושמה Function_Table_entry. המחלקה הנ"ל מכילה מידע על הפונקציות כמו:

- היכן הפונקציה הוגדרה בקוד (באיזו שורה).
- מהו סוג החזרה של הפונקציה (int,float,void)
- שם הפונקציה
- וקטור שמכיל את הפרמטרים שהפונקציה מקבלת
- וקטור שמכיל מספרי שורות בבאפר שקוראות לפונקציה.

בנוסף למפה הנ"ל למחלקה של טבלת הסמלים לפונקציות הוספנו גם מידע על הפונקציה שנמצאת כרגע בביצוע. עשינו זאת עבור המקרה בו יש return ועלינו לאכוף כי ערך טיפוס החזרה מתאים לטיפוס החזרה כפי שצוין בעת הגדרת הפונקציה.



<u>טבלת סמלים למשתנים:</u> את הטבלה הנ"ל בנינו באופן ההיררכי הבא:



למעשה טבלת הסמלים למשתנים הגלובלית מכילה בתוכה **רשימה של טבלאות סמלים עבור scope** וזאת כדי שנוכל לעקוב מאיזה scope מגיע כל משתנה שיש בו שימוש. כל טבלת משתנים עבור scope מכילה **שתי טבלאות משתנים קטנות יותר עבור כל טיפוס.**

למשל אם הצהרתי על משתנה x מסוג int בתוך מימוש של פונקציה בשם foo. המשתנה הנ"ל יישב בתוך טבלת הסמלים הגלובלית בתוך טבלת ה-scope של foo. בתוך טבלת ה-scope הוא יישב בטבלה של משתני ה-int.

מעקב אחר תחת איזה scope יושב כל משתנה מאפשר לנו, ככותבי התוכנית ב—Cלהשתמש באותו שם משתנה בסקופים שונים ללא קונפליקט לאיזה scope אנחנו מתכוונים.

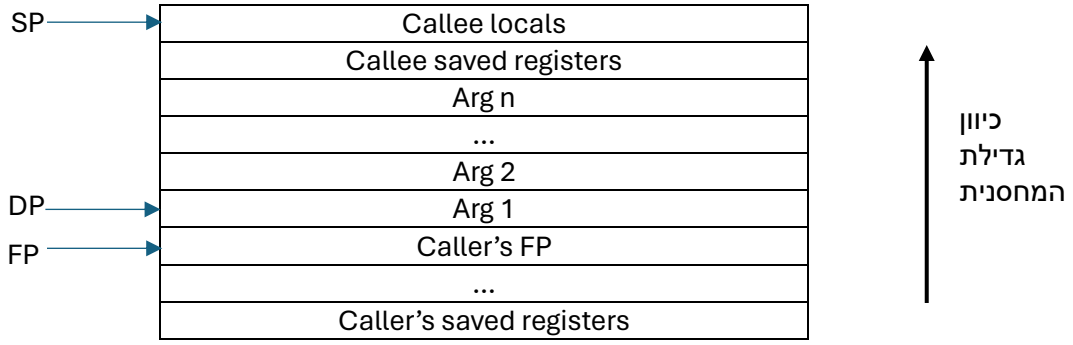
<u>אופן הקצאת הרגיסטרים השמורים:</u>

להלן פריסת הרגיסטרים המיוחדים :

שם הרגיסטר	סימון ב-INT	סימון ב-FLOAT	תכונה
RA	I0	אין	מחזיק את כתובת החזרה לאחר סיום פונקציה
SP	I1	F0	מחזיק את מיקום ראש המחסנית
FP	I2	F1	מחזיק את מיקום תחילת רשומת ההפעלה הנוכחית
DP	I3	F2	מחזיק את מיקום הפרמטר הראשון לפונקציה ברשומת ההפעלה
RT	I4	F3	מחזיק את ערך החזרה של הפונקציה

מבנה רשומת ההפעלה:

להלן מבנה רשומת ההפעלה כפי שמומש בקומפיילר:



- בכל קריאה לפונקצייה, ה-Caller שומרת את הרגיסטרים שלה על המחסנית וכמו כן שומרת את הפרמטרים של ה-Callee על המחסנית, לשם מצביע הרגיסטר DP.
- הרגיסטר FP מציין היכן מתחילה הרשומה של הCallee.
- כל פעם ש-Callee מקצה משתנה זמני/רגיסטר המחסנית גדלה כלפי מעלה והמשתנה הנ"ל נשמר על גביה.
- בחזרה מקריאה, ה-Caller משחזר את הרגיסטרים ששמר על המחסנית.

המודולים השונים בקוד:

הקוד שלנו מורכב מחמשת הקבצים הבאים:

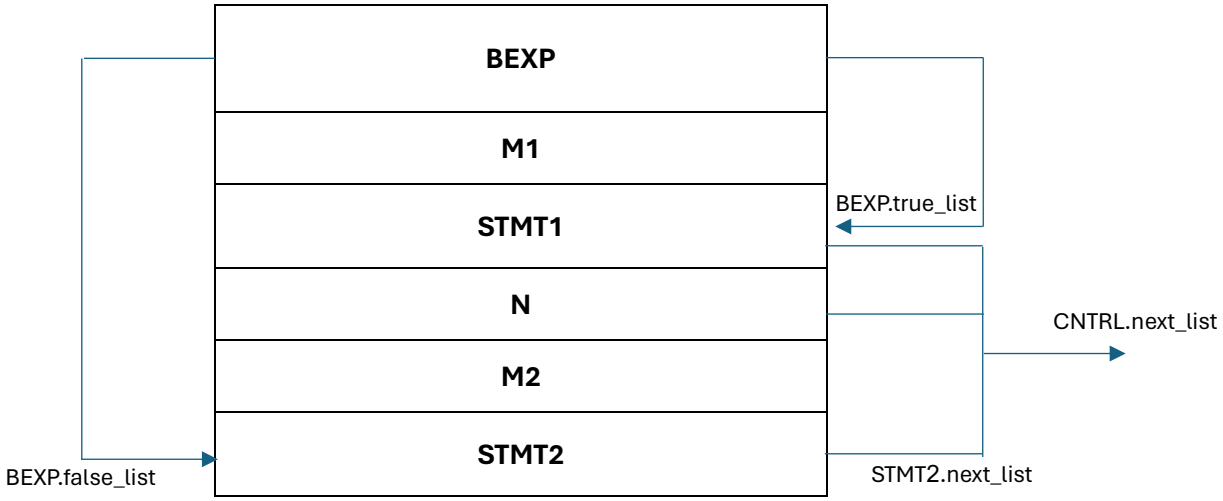
- Lexer.lex – קובץ הלקסר שבנינו בחלק 1 של הפרויקט.
- Parser.ypp – קובץ הפרסר שבנינו בחלק 2 של הפרויקט לאחר שהוספנו לו את כל הבדיקות הסמנטיות, עבודה מול טבלאות סמלים וייצור קוד שלשות.
- helpers.cpp – קובץ המכיל את המימוש של כל מחלקות העזר, כל פונקציות העזר, טבלאות הסמלים.
- helpers.h - קובץ המכיל הצהרות על כל המחלקות שהשתמשנו בהם בפרויקט, טבלאות הסמלים.
- Makefile

פריסת הקוד למבני הבקרה:

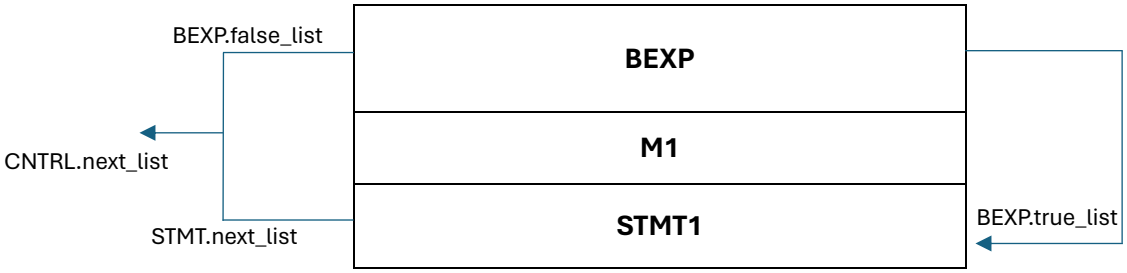
נציין מהי פריסת הקוד עבור שלושת מבני הבקרה הבאים:

- CNTRL → IF BEXP THEN M1 STMT ELSE N M2 STMT
- CNTRL → IF BEXP THEN M STMT
- CNTRL → WHILE M1 BEXP DO M2 STMT

עבור **CNTRL → IF BEXP THEN M1 STMT ELSE N M2 STMT** להלן מבנה הבקרה:



עבור **CNTRL → IF BEXP THEN M STMT** להלן מבנה הבקרה:



עבור **CNTRL → WHILE M1 BEXP DO M2 STMT** להלן מבנה הבקרה: