# פרויקט - חלק 3

## תיעוד הפרויקט

מגישים: בר דוד ושיר פן

##הסבר כללי על מימוש הקומפיילר

הקומפיילר מבצע תרגום של קוד משפת C—לשפת RISKI.

המימוש מבוסס על ארכיטקטורת LR- bottom-up ומשתמש בכלים bison,flex.

התרגום מתבצע באופן מונחה תחביר ושימוש בשיטת backpatching. עבור כל כלל גזירה שמופעל, נוספות פקודות מתאימות ל-buffer (משתנה גלובלי) אשר משמשות בהמשך ליצירת קובץ הריצה.

##מבני הנתונים המשמשים לניהול מצב הקומפיילר

1. טבלת סמלים

טבלת הסמלים מיושמת כמבנה נתונים מסוג map, כאשר המפתח הוא שם המשתנה והערך הוא אובייקט מסוג Symbol.

אובייקט זה מכיל:

* וקטור טיפוסים: מכיל את כל הטיפוסים של המשתנה בהגדרותיו השונות
* וקטור היסטים: מציין את המיקום בזיכרון ביחס לתחילת מקטע הנתונים
* עומק: מציין את רמת הבלוק העמוקה ביותר בה הוגדר המשתנה

בעת הגדרת משתנה חדש, המערכת בודקת האם קיים כבר משתנה באותו שם ובאותו בלוק. הבדיקה מתבצעת על ידי השוואת רמת הבלוק הנוכחית עם שדה העומק בטבלת הסמלים. אם קיים משתנה כזה, מופקת שגיאה. אחרת, נוספת הגדרה חדשה לטבלת הסמלים.

1. טבלת פונקציות

טבלת הפונקציות היא מבנה נתונים גלובלי מסוג map, כאשר המפתח הוא שם הפונקציה והערך הוא אובייקט מסוג Function, אובייקט Function מכיל:

* כתובת המימוש של הפונקציה (אם קיים)
* טיפוס ערך החזרה
* וקטור של טיפוסי הפרמטרים
* וקטור של כתובות הקריאה בקובץ הriski
* דגל המציין האם הפונקציה מומשה
* דגל המציין האם הפונקציה היא וואריאדית
* משתנה המחזיק את האינדקס של המשתנה הוואריאדי הראשון (במידה והפונקציה אינה וואריאדית מחזיק ערך של 0).
* משתנה המחזיק את האינדקס של הארגומנט בו מועברים מספר המשתנים הוואריאדים של הפונקציה, מוגדר להיות אינדקס הארגומנט האחרון לפני '...' (במידה והפונקציה אינה וואריאדית מחזיק ערך של 0).
* משתנה המחזיק את מספר הארגומנטים הוואריאדים שמועברים לפונקציה, מאותחל ל 1- ומשתנה בקריאה לפונקציה וואריאדית בהתאם לארגומט הרלוונטי.

##מימוש פונקציות עם פרמטרים משתנים (VA)

המימוש של פונקציות עם פרמטרים משתנים כולל מספר מרכיבים:

1. זיהוי פונקציה כוואריאדית: פונקציה מזוהה כבעלת פרמטרים משתנים כאשר היא מוגדרת עם (...) בסוף רשימת הפרמטרים.

2. ניהול מידע על פרמטרים משתנים:

-startIndexVariadicParams מציין את האינדקס בו מתחילים הפרמטרים המשתנים

-indexNumVAParamsמציין את האינדקס של הפרמטר המכיל את מספר הפרמטרים המשתנים

- numVariadicParamsמספר הפרמטרים המשתנים בפועל

3. בדיקות בזמן קומפילציה:

- וידוא שהפרמטר האחרון הקבוע הוא מטיפוס Int (מייצג את מספר הפרמטרים המשתנים)

- בדיקת תאימות טיפוסים עבור הפרמטרים הקבועים

4. מימוש va\_arg:

- חישוב ההיסט בזיכרון לפרמטר המבוקש

- בדיקות תקינות בזמן ריצה (האם האינדקס בטווח החוקי)

- טעינת הערך מהזיכרון בטיפוס המבוקש

5. בדיקות בזמן ריצה:

- ווידוא בדיקת תקינות השימוש בva\_arg בהתאם לכמות הפרמטרים הוואריאדים (כפי שתואר ב4)

- ווידוא שמספר הפרמטרים המועברים בפועל שווה למספר המועבר בפרמטר במיקום indexNumVAParams.

##תיאור ה-Backpatching לכל מבני הבקרה:

1. תנאי if:

כתובת ה true של הביטוי הבוליאני מוטלאת לתחילת גוף הif.

כתובת הfalse מאוחדת עם כתובת הnext של הגוף ומועברת לטיפול בשלב הבא של הreduce.

1. תנאי if-else:

דומה למבנה הif הפשוט, אך כתובת הfalse של הביטוי הבוליאני מוטלאת ישירות לגוף הelse במקום להתאחד עם הnext.

1. לולאת while-do:

- הטלאה לאחור של כתובת הtrue אל גוף הלולאה

- כתובת הfalse מועברת לnextList של הreduce הבא

- כתובת הnext של גוף הלולאה מוטלאת לתחילת הביטוי הבוליאני

##מבנה רשומת ההפעלה

השתמשנו בהמצלה מהוראות התרגיל ולכן רשומת ההפעלה מורכבת מהחלקים הבאים:

A screenshot of a computer screen

AI-generated content may be incorrect.

* תפקידי הפונקציה הקוראת:

בקריאה:

- שמירת רגיסטרים פעילים

- העברת פרמטרים לפי הסדר בהגדרה של הפונקציה

- הקצאת מקום לערך החזרה

- עדכון רגיסטרים I1,I2

-אם הפונקציה וואריאדית, בדיקת נכונות הקריאה לפונקציה (בזמן ריצה)

- ביצוע קריאה באמצעות JLINK

-שמירת הכתובת לחזרה של הפונקציה הנקראת (השורה הבאה)

בחזרה:

-שחזור הרגיסטרים I1,I2 לערכם המקורי

-שחזור כל הרגיסטרים הפעילים מהמחסנית (אלו ששמרנו לפני הקריאה לפונקציה)

-שמירת ערך החזרה מהפונקציה הנקראת

* תפקידי הפונקציה הנקראת:

בקריאה:

- הרצת הקוד המתורגם מקובץ הInput של הקומפיילר (כלומר קובץ הcmm)

- שימוש בפרמטרים מכתובות I1 -4 ומטה

- אפשרות להקצאת זיכרון מ-I2 ומעלה

בחזרה:

-שמירת ערך החזרה בהיסט של I1 -4

## אופן הקצאת רגיסטרים מיוחדים (שמורים)

רגיסטרים שמורים:

1. ערך החזרה מפונקציה- I0
2. Frame pointer- I1
3. Stack pointer- I2

רגיסטרים זמניים מוקצים החל מרגיסטר 3, כל המשתנים בעלי גודל של 4 בתים ולכן חישוב ההיסט הוא תמיד בכפולות שלמות של 4.

## תיאור המודולים השונים בקוד של הקומפיילר

1. part3.ypp

קובץ הbison הראשי המכיל את כללי הדקדוק והפעולות הסמנטי המבצע ניתוח תחבירי ותרגום לקוד riski על ידי הקצאת buffer והדפסת פקודות הriski לתוכו.

2. part3\_helper.hpp

- הגדרות מבני הנתונים

- הצהרות על פונקציות עזר

- קבועים וטיפוסים בסיסיים

3. part3\_helpers.cpp

- מימוש פונקציות העזר מקובץ ה part3\_helper.hpp

4. part3.lex

- מנתח לקסיקלי מבוסס flex

- מזהה טוקנים בקוד המקור