פרויקט – חלק 3
תיעוד הפרויקט

נועה בינבאום

- **הסבר כללי על מימוש הקומפיילר**

הקומפיילר שבנינו מבצע תרגום של קוד מהשפה --C לשפה Riski. הקומפיילר הוא LR מסוג bottom up המבוסס על שימוש בכלים bison ו-flex ומיישם תרגום מונחה דקדוק כך שהוא מבצע את התרגום בשיטת backpatching. בכל פעם שמופעל כלל גזירה, מודפס ל-buffer (משתנה גלובלי עליו יפורט בהמשך) פקודות המאפשרות להמשך ביצוע שלב הלינקר והפקת קובץ ריצה.

- **מבני נתונים המשמשים לניהול מצב הקומפיילר**

1. טבלת סמלים:

משתנים בקוד מוכנסים לטבלת סמלים גלובלית ברגע הכרזתם. טבלת הסמלים היא מסוג map שבה ה-key הוא שם הסמל והערך הינו מבנה נתונים מסוג Symbol (בעל השדות: וקטור טיפוסים עבור כל ההגדרות של אותו משתנה, וקטור היסטים ביחס לתחילת זיכרון הנתונים בו שמורה ההגדרה של כל משתנה, ועומק, המציין מהי רמת העומק הגבוהה ביותר של בלוק בו הוגדר משתנה בשם זה). בכל פעם שמגדירים משתנה חדש, בודקים האם קיים כבר משתנה באותו שם ובאותו בלוק (משווים את גודלו של משתנה גלובלי המחזיק ברמת הבלוק הנוכחית בה אנו נמצאים לשדה depth של המשתנה בטבלת הסמלים). במידה וכבר קיים משתנה בשם זה באותה רמה, תוחזר שגיאה. אחרת, מעדכנים בטבלת הסמלים כי קיימת הגדרה חדשה של המשתנה באותו בלוק. באופן זה, בכל פעם שיוצאים מבלוק, מוחקים מווקטורי ההיסט והטיפוס של כל המשתנים את האיבר שהוכנס באותה רמה, ובכך מוחקים את רישום המשתנים שהוגדרו בתוך הבלוק, בעת היציאה מהבלוק ובנוסף שומרים בכל רמה את כל ההגדרות שבוצעו לכל המשתנים ברמות העמוקות יותר. כך ניתן לגשת להגדרה הכי עמוקה של כל משתנה מתוך כל בלוק פנימי יותר לזה שבו בוצעה הגדרת המשתנה.

2. טבלת פונקציות:

פונקציות בקוד נשמרות בטבלה גלובלית. טבלה זו הינה מבנה נתונים מסוג map שבו ה-key הוא שם הפונקציה וה-value הוא מבנה נתונים בשם Function.
מבנה הנתונים Function מכיל את כתובת המימוש של הפונקציה (במידה וקיים מימוש באותו קובץ), טיפוס ערך החזרה של הפונקציה, וקטור טיפוסי הפרמטרים של הפונקציה, וקטור כתובות בקובץ rsk. בו מתבצעות קריאות לפונקציה, אינדיקטור האם הפונקציה מומשה (כדי למנוע מימוש כפול). בכל פעם שמבצעים הצהרה של פונקציה, בודקים האם כבר הוצהרה והוגדרה כיוון שלא ניתן לבצע ריבוי הצהרות והגדרות של פונקציות. בנוסף נבדק כי ערך ההחזרה, סוגי הפרמטרים ומספרם תואם בהגדרה להצהרה של הפונקציה.

3. מבנה הנתונים YYSTYPE:

מבנה זה מכיל את כל התכונות אותן יש להעביר בין משתני השפה בעת ביצוע reduce לפי כללי הגזירה. התכונות הינן:

- שם המשתנה (במידה ומדובר ב-id או ב-num אז תכונה זאת מכילה את שם המשתנה/ הפונקציה או ערך מספרי של num)
- טיפוס
- היסט ביחס לזיכרון הנתונים
- רשימות nextlist, true list, falselist, המשמשות לטובת הטלאה כפי שנלמד בתרגולים ובהרצאות
- כתובת יחסית לקובץ תוצר הקומפילציה (קובץ rsk.)
- מספר רגיסטר מתוך קובץ הרגיסטרים (החל מרגיסטר מספר 3)
- וקטור טיפוסים של פרמטרים לפונקציה
- וקטור היסטים של פרמטרים לפונקציה ביחס לזיכרון
- וקטור מספרי רגיסטרים של פרמטרים לפונקציה

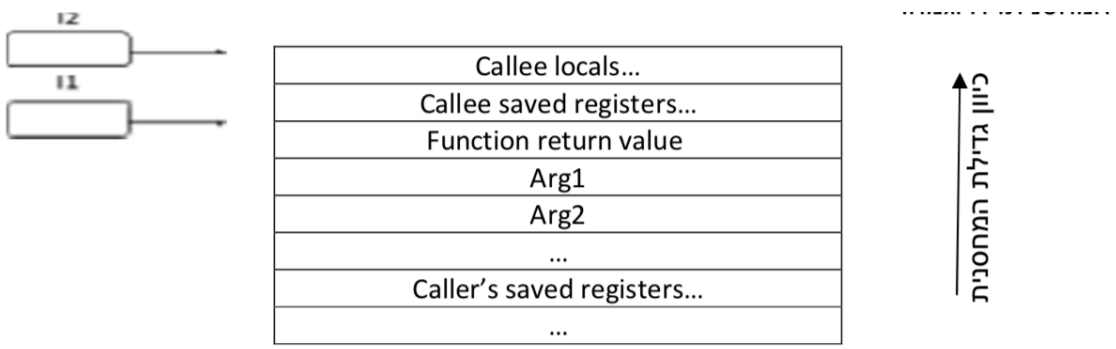
- **תיאור ה-backpatching לכל אחד ממבני הבקרה**

1. תנאי if: כתובת ה-TRUE של הביטוי הבוליאני מוטלאת אל תחילת גוף ה-if ואילו כתובת ה-FALSE מאוחדת עם כתובת ה-next של הגוף ומועברת לטיפול ע"י ה-reduce הבא כאשר נדע את הקוד של כל מבנה הבקרה.
2. תנאי if-else: באופן זהה למבנה הקודם, עם ההבדל שכתובת ה-FALSE של הביטוי הבוליאני לא מאוחדת עם ה-next אלא מוטלאת לגוף של ה-else.
3. לולאת while: בעת ביצוע reduce, מתבצעת הטלאה לאחור של כתובת ה-TRUE שך הביטוי הבוליאני אל תוך גוף הלולאה, ואילו כתובת ה-FALSE מועברת לטיפול ע"י ה-nextList של ה-reduce הבא שכן יש לקפוץ לכתובת שאחרי גוף הלולאה כולה. כתובת ה-next של גוף הלולאה מוטלאת לכתובת ההתחלה של הביטוי הבוליאני לשם הערכת התנאי.

4. יציאה מבלוק: בעת ביצוע reduce בכלל מתוכן הבלוק למילה BLK מבצעים הטלאה לאחור של ה-next בגוף הבלוק לשורה ב-buffer לאחר גוף הבלוק
5. קריאה לפונקציות: בעת ביצוע כלל הגזירה האחרון בתכנית (הנגזר למשתנה התחילי בשפה- PROGRAM) כבר יודעים את כתובות תחילת המימוש של כל הפונקציות הממומשות בקובץ ולכן מבצעים הטלאה לאחור עבור כל פונקציה בכל השורות בהן הפונקציה נקראת (לפי השדה callingLines של כל פונקציה בטבלת הפונקציות).

- **מבנה רשומת ההפעלה**

הגדרת רשומת ההפעלה נעשה כפי שהומלץ בהוראות התרגיל:



תפקיד הפונקציה הקוראת בקריאה לפונקציה:

- שמירת כל הרגיסטרים המוקצים בזיכרון באופן רציף
- שמירת כל הפרמטרים לפונקציה הנקראת לפי סדר הופעתם בהגדרת הפונקציה
- שמירת מקום בזיכרון לערך החזרה של הפונקציית הנקראת
- עדכון רגיסטר I2 לגודל הזיכרון הנוכחי
- עדכון I1=I2
- שמירת ערך ה-PC ב-I0 וקריאה לפונקציה באמצעות JLINK

תפקיד הפונקציה הנקראת בקריאה לפונקציה:

- לא עושה כלום מלבד הרצת הקוד הנוצר ממימוש הפונקציה בקובץ ה-cmm. כאשר ידוע לה שהפרמטרים שלה מצויים בכתובת I1-4 ומטה והיא יכולה להקצות זיכרון מ-I2 ומעלה

תפקיד הפונקציה הנקראת בחזרה מהפונקציה:

- השמת ערך החזרה בכתובת I1-4

תפקיד הפונקציה הקוראת בחזרה מהפונקציה:

- שחזור ערך ה-SP: I2=I1
- שמירת ערך החזרה לרגיסטר
- שחזור ערכו הקודם של I1
- טעינת כל הרגיסטרים מתוך המחסנית בזיכרון לפי סדר הכנסתם

- **אופן הקצאת רגיסטרים מיוחדים (שמורים)**

הרגיסטרים השמורים בתכנית הינם:

1. I0 – כתובת החזרה מפונקציה
2. Frame Pointer – I1
3. Stack Pointer – I2

הקצאת רגיסטרים זמניים (לשימוש התכנית נעשית החל מרגיסטר מספר 3).
בכל פעם שפונקציה נקראת מבצעים שמירה של הרגיסטרים בזיכרון באופן שבו הם מיושרים לתחילת הזיכרון, כל משתנה בגודל 4 בתים יישמר בכתובת שהינה כפולה של 4, משתנה בגודל 2 בתים בכתובת שהינה כפולה של 2 ומשתנה בגודל בית אחד בכתובת הפנויה הבאה בזיכרון. בכל פעם כשצריך לבצע STI בזיכרון נבדק גודל הטיפוס שנשמר וגודל ההיסט הנוכחי בזיכרון. תחילה גודל ההיסט משתנה בהתאם לצורך ביישור אותו טיפוס בזיכרון כפי שהוסבר, ולאחר מכן ההיסט גדל באותו מספר בתים כגודל הטיפוס המתווסף לזיכרון.

- **תיאור המודולים השונים בקוד של הקומפיילר**

1. קובץ Lex: המנתח הלקסיקלי המבוצע ע"י flex
2. קובץ part3_helper.hpp: מכיל את הגדרת מבני הנתונים המשמשים לבניית הקומפיילר
3. קובץ part3_helper.cpp: מכיל את מימושי הפונקציות המוצהרות בקובץ part3_helper.hpp
4. part3.ypp: קובץ ה-bison והתכנית הראשית של הקומפיילר. תפקיד ה-bison הוא לבצע ניתוח תחבירי ופירוש סמנטי. התכנית הראשית מקצה את ה-buffer לכתיבת הקוד, יוצרת קובץ rsk. בהתאם לתכנית הכתובה בקובץ cmm. ומדפיסה לתוכו את כל הקוד הנוצר (תוכן ה-buffer).