מבוא למערכות מחשוב

**עבודה מספר 4 – קומפילציה**

**כללי** – בעבודה זאת תכתבו קומפיילר פשוט המתרגם משפטי השמה בפונקציה לקוד אסמבלי. משפטי ההשמה יוכלו לקבל ביטויים מורכבים, ולכן יש צורך בפישוט הביטויים באמצעות משתנים לוקליים. הקוד המתקבל בסיום הריצה יוכל לעבור תרגום לשפת מכונה על ידי האסמבלר, ואח"כ לרוץ על המכונה שבניתם.

למען הפשטות, כל הביטויים יהיו בתצורת prefix המאפשרת עבודה על פי דקדוק LL(0), כלומר קבלת החלטות על פי הסמל (token) הראשון. דוגמאות לביטויים מתצורת prefix:

5

(+ 4 5)

(- x 3)

(+ 3 (+ x y))

(- (+ 3 4) (+ 2 1))

הביטויים לא יכללו קריאות לפונקציות (לדוגמה f(x)), והפעולות היחידות המותרות יהיו פעולות חיבור וחיסור. כל הפעולות יהיו בינאריות, כלומר בעלות שני אופרנדים בלבד.

עליכם לממש את החלקים השונים בשיטה Compile המקבלת מחרוזת המייצגת משפט השמה, ומקמפלת אותו לקוד אסמבלי. כל משפטי ההשמה יהיו מהתצורה:

let <variable name> = <expression>;,

כאשר expression הוא ביטוי מתצורת Prefix כפי שהוסבר להלן, ו-variable name הוא שם חוקי עבור משתנה.

**שלבי הביצוע:**

1. ממשו את השיטה Tokenize המקבלת מחרוזת ומחזירה מחסנית של סמלים (tokens) המסודרים בסדר הפוך, כלומר, הסמל הראשון יהיה בראש המחסנית. עבור כל Token יש לשמור את מספר השורה בה הוא הופיע, את מיקומו בשורה, את שמו, ואת הסוג שלו. דוגמאות:
   1. let x = (+ 41 52); יחזיר את המחסנית הבאה (ראש המחסנית משמאל, ואיבריה מופרדים על ידי פסיקים): let,x,=,(,+,41,52,),;
   2. let y =(+ (- 4 (+ 3 2)) 6); => let,y,=,(,+,(,-,4,(,3,2,),),6,),;
2. ממשו את השיטה Parse המקבלת את מחסנית ה-tokens ומייצרת אובייקט מטיפוס AssignmentStatement תוך שימוש במחלקות השונות היורשות את Expression:
   1. NumericExpression – מייצג ביטוי מספרי, כגון 7
   2. VariableExpression – מייצג משתנה, כגון x
   3. BinaryOperationExpression – מייצג פעולה (+,-) על שני אופרנדים מטיפוס expression. באופן זה ניתן לבנות ביטויים מכוננים.
   4. שימו לב – ה-ToString של ה-AssignmentStatement שתיצרו אמור להיות זהה למחרוזת המקורית שקיבלתם כקלט (למעט הסרת מספר whitespaces עוקבים).
   5. במהלך ה-Parsing יתכן ויתגלו טעויות תחביר (syntax). במקרים כאלו יש לזרוק SyntaxErrorException המכיל מידע על התקלה ואת ה-Token בו התרחשה התקלה. למשל:

throw new SyntaxErrorException(“Expecting ; at the end of the statement”, t)

1. ממשו את השיטה SimplifyExpression המקבלת משפט השמה עם ביטוי מורכב ומפשטת אותו לסדרה של משפטי השמה פשוטים, כך שכל משפט השמה יכיל מספר, משתנה, או פעולה בינארית על משתנים או מספרים, ללא ביטויים מכוננים. לשם כך נגדיר משתנים מלאכותיים, שיקראו על פי המוסכמה “\_i” כאשר i הוא המספר הסידורי של המשתנה המלאכותי. גם משתנים אלו יש להכניס לתוך טבלת הסמלים.

לדוגמא: במשפט "let x = (+ (- 53 12) (- 467 3));" יצור את שרשרת המשפטים:

let \_1 = (- 53 12);

let \_2 = (- 467 3);

let \_3 = (+ \_1 \_2);

let x = \_3;

1. ממשו את השיטה ComputeSymbolTable, המקבלת את עצי הגזירה, כולל הביטויים הפשוטים, שחושבו בסעיפים הקודמים, ויוצרת טבלת סמלים. בטבלה שלנו יש רק לשמור את שם המשתנה ואם האינדקס שלו. אין צורך לשמור טיפוס, מאחר ונשתמש רק ב-Int, ואין צורך לשמור סוג מאחר ונשתמש רק ב-local.
2. ממשו את השיטה GenerateCode המקבלת משפט השמה פשוט (כלומר, מכיל רק ביטוי פשוט יחיד) וממירה אותו לקוד אסמבלי. שימו לב, זוהי אינה משימה פשוטה, שכן עלינו לתמוך בפעולות על משתנים הנמצאים בהיסט מסוים מ-LOCAL. נשתמש לכן בשלבים הבאים:
   1. כאשר מחשבים ביטוי המורכב מפעולה חישובית, נעביר ראשית את שני האופרנדים (או כתובותיהם – לבחירתכם) אל שני רגיסטרי אסמבלי שיקראו OPERAND1 ו-OPERAND2.
   2. חישוב הביטוי בהשמה לתוך RESULT (רגיסטר אסמבלי מיוחד).
   3. העברת הערך מ-RESULT אל המשתנה הלוקלי.
   4. רצוי לכתוב פונקציות גנריות כגון CopyToOperand1 במקום לשכפל את הקוד פעמים רבות.

להזכירכם, העבודה אינה בזוגות. כל העברה של קטעי קוד בין סטודנטים אסורה בהחלט. כל צורה של העתקה תטופל בחומרה המירבית – אל תעתיקו!

את העבודה יש להגיש עד ליום ראשון, ה-17 בינואר, בשעה 23:55. לעבודה זאת לא יהיו הארכות!