

מבחן סוף סמסטר – מועד א' - פתרון

מרצה אחראי:

דר' ערן ירב

מתרגלים:

רן צמח

עדי סוסנוביץ'

רן בן-בסט

פתרון שאלה 1

- א. (2 נק') בקריאה לפונקציה $x = \text{foo}()$ נמצא שהטיפוס של x אינו תואם את טיפוס ערך החזרה של הפונקציה. זמן קומפילציה. הטיפוס של ערך חזרה מפונקציה $\text{foo}()$ והטיפוס של המשתנה x נשמרים כחלק מהמידע בשלב הניתוח הסמנטי. בדיקת התאמה בין הטיפוסים מתבצעת בשלב ה type checking שם תדווח שגיאה במקרה של אי התאמה.
- ב. (2 נק') נמצא שהדקדוק עבור שפת התכנות הוא רב-משמעי. זמן בניית הקומפיילר. שימוש בכלי לייצור parser כמו yacc ידווח על שגיאה במקרה שהדקדוק הוא רב-משמעי.
- ג. (2 נק') ניתן להקצות את המשתנה המקומי x והמשתנה המקומי y לאותו רגיסטר. זמן קומפילציה. שלב $\text{register allocation}$. ניתוח בזמן קומפילציה יגלה שהמשתנים לא חיים סימולטנית ולכן ניתן להקצות אותם לאותו הרגיסטר.
- ד. (2 נק') השם $\$aloha$ אינו שם משתנה חוקי. זמן קומפילציה, למרות שהחלטה על כך ששמות משתנים כאלה אינם חוקיים מתבצעת כבר בזמן בניית הקומפיילר.
- ה. (2 נק') פעולת החלוקה $z = x / y$ גורמת לחילוק באפס. באופן כללי בזמן ריצה. במקרים מסוימים בהם הערך y מובטח להיות קבוע וניתן לגלות זאת בזמן קומפילציה, אפשר לדווח על השגיאה בזמן קומפילציה. ברוב המקרים הערך של y יהיה ידוע רק בזמן ריצה. שפה שמבצעת בדיקה של ערך המכנה לפני חלוקה בדרך כלל תזרוק exception במקרה זה.

פתרון שאלה 2

(א) זרימה קדמית, בעיית may, פריטי מידע: {LOCK, NO_LOCK}

$$gen(B) = \begin{cases} \{LOCK\} & , B \text{ is a LOCK command} \\ \{NO_LOCK\} & , B \text{ is an UNLOCK command} \\ f & , B \text{ is any other command} \end{cases}$$

$$kill(B) = \begin{cases} \{NO_LOCK\} & , B \text{ is a LOCK command} \\ \{LOCK\} & , B \text{ is an UNLOCK command} \\ f & , B \text{ is any other command} \end{cases}$$

אתחול: NO_LOCK לכל הבלוקים.

אופן מתן אזהרה: על כל בלוק של פקודת LOCK, שבו IN(B) מכיל LOCK.

(ב) זרימה קדמית, בעיית may, פריטי מידע: {UNLOCK, NO_UNLOCK} סימטרי ל-א' בפונקציות gen, kill.

אופן מתן אזהרה: על כל בלוק של פקודת UNLOCK, שבו IN(B) מכיל UNLOCK.

(ג) זרימה אחורית, בעיית may, פריטי מידע: {EXITS, CANNOT_EXIT} אתחול: in[exit] = EXITS, בלוקים אחרים מאותחלים עם CANNOT_EXIT.

$$gen(B) = \begin{cases} \{CANNOT_EXIT\} & , B \text{ is an UNLOCK / LOCK command} \\ f & , B \text{ is any other command} \end{cases}$$

$$kill(B) = \begin{cases} \{EXITS\} & , B \text{ is an UNLOCK / LOCK command} \\ f & , B \text{ is any other command} \end{cases}$$

אופן מתן אזהרה: על כל בלוק של פקודת LOCK, שבו OUT(B) מכיל EXITS.

(ד) זרימה קדמית, בעיית may, פריטי מידע: {ENTERS, CANNOT_ENTER} אתחול: out[Entry]=ENTERS, בלוקים אחרים מאותחלים עם CANNOT_ENTER

$$gen(B) = \begin{cases} \{CANNOT_ENTER\} & ,B \text{ is an UNLOCK / LOCK command} \\ f & ,B \text{ is any other command} \end{cases}$$

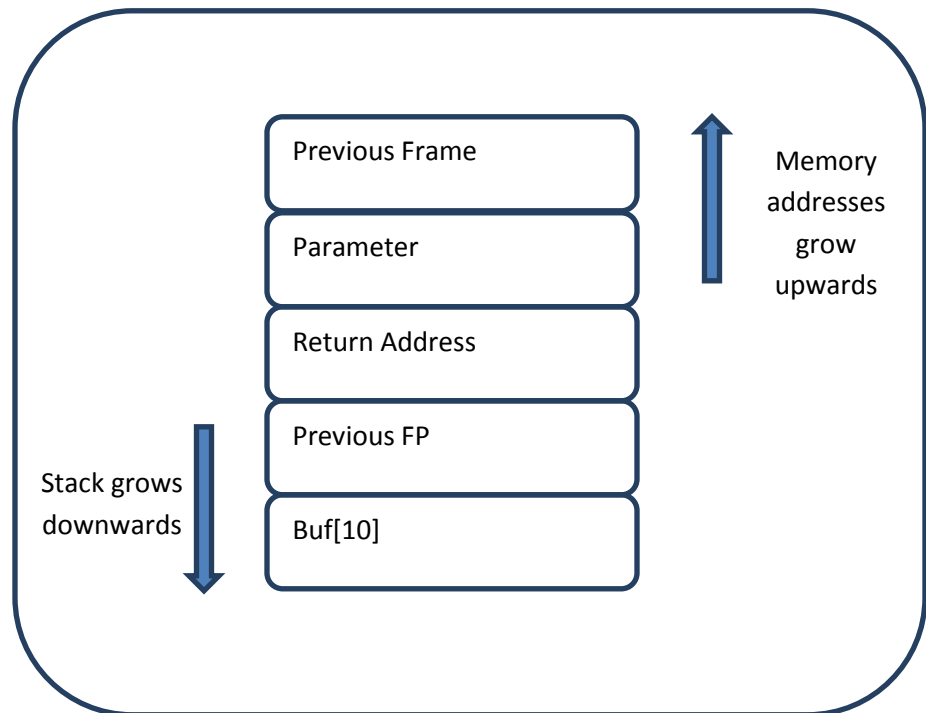
$$kill(B) = \begin{cases} \{ENTERS\} & ,B \text{ is an UNLOCK / LOCK command} \\ f & ,B \text{ is any other command} \end{cases}$$

אופן מתן אזהרה: על כל בלוק של פקודת UNLOCK, שבו IN(B) מכיל ENTERS.

פתרון שאלה מספר 3

א. (5 נק') הסבירי את מבנה רשומת ההפעלה בקריאה לפונקציה foo בדוגמא ומדוע קלט באורך של יותר מ 11 תווים ידרוס את כתובת החזרה. ניתן, ואף רצוי, להדגים את מבנה רשומת ההפעלה בתרשים.

לשם פשטות ההצגה בכל הפתרון נניח שאין שמירה של רגיסטרים במחסנית, לא callee save ולא caller save. מבנה רשומת ההפעלה הוא



כאשר כותבים לתוך המערך יותר מ 10 תווים, הערכים יתחילו לגלוש על תאים קודמים של המחסנית. תחילה ידרס הערך הקודם של FP ואחריו תדרס כתובת החזרה.

דריסה של כתובת החזרה תגרום לכך שהשגרה foo לא תחזור לכתובת המקורית אליה היא אמורה לחזור. על ידי בחירה של כתובת חזרה מסוימת, ניתן לגרום לתוכנית לחזור לקוד שלנו ולבצע קוד זדוני כלשהו.

ברצוננו לממש מנגנון אשר יבדוק דריסה של כתובת החזרה בזמן ריצה. כלומר, כאשר מריצים את התכנית לעיל עם קלט ארוך מ 11 תווים, תתקבל הודעת שגיאה בזמן ריצה. על הקומפיילר לייצר את הקוד המתאים לצורך כך.

ב. (10 נק') הסבירי את השינויים הנדרשים בקומפיילר ותארי את הפרטים הבאים:

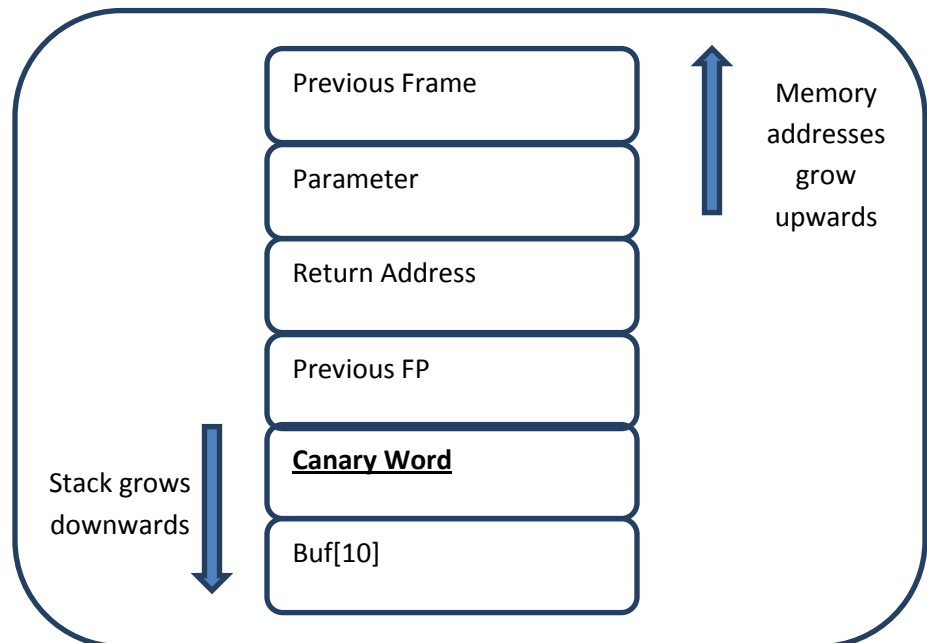
- השינויים הדרושים ברשומת ההפעלה. יש לתאר במפורט את המבנה החדש של רשומת ההפעלה. ניתן להדגים זאת בתרשים.
- השינויים הנדרשים בקוד שמנהל את רשומות ההפעלה. יש לתאר במפורט את המבנה החדש של הקוד ברמה של פעולות מכונה. **אין צורך** להשתמש בסינטקס של שפת מכונה מסוימת.

הדגימי כיצד השינויים יאפשרו את גילוי השגיאה בדוגמא שלנו.

פתרון:

רעיון: נוסיף לרשומת ההפעלה מילה בעלת תוכן אקראי כלשהו אשר תפריד בין כתובת החזרה לבין שאר תוכן המחסנית. מובטח לנו שכל דריסה רציפה שמקורה במשתנים הלוקאליים ותדרוס את כתובת החזרה תדרוס לפני כן את המילה האקראית שלנו. לפני החזרה מהפונקציה, נבדוק את תוכן המילה האקראית, אם היא נדרסה, נזרוק שגיאת זמן ריצה. אם התוכן שלה חוקי, גם כתובת החזרה לא נדרסה וניתן לחזור מהשגרה בבטחה.

עבור מילה בת 64 ביט, הסיכוי שתוקף יוכל לנחש את התוכן האקראי שלנו הוא זניח.



פרטים:

Original prolog for foo:

```

pushl   %ebp                # save old ebp
movl    %esp, %ebp         # bp = sp
subl    $56, %esp          # allocate stack space for locals

```

Modified prolog sequence:

```

pushl   %ebp                # save old ebp
pushl   canary-word        # randomly generated canary
movl    %esp, %ebp         # bp = sp
subl    $56, %esp          # allocate stack space for locals

```

באופן דומה, יש לשנות את האפילווג של הפונקציה כך שיבדוק את חוקיות הקנרית לפני החזרה.

פתרונות אחרים ומדוע חלק מהם לא עובדים

פתרון 1: שמירת כתובת החזרה ברגיסטר

הפתרון הזה שומר את ערך החזרה של השגרה האחרונה. לא ברור איך הוא מתעדכן כאשר חוזרים מהשגרה לשגרה שקראה לה. הרי קוד של קריאה חדשה יצטרך לשמור את ערך הרגיסטר הזה במקום כלשהו בזכרון. אפשר לגרום לפתרון הזה לעבוד, אבל רוב הפתרונות שראיתי לא באמת גרמו לו לעבוד והחסירו הרבה פרטים כאשר הפרטים כאן לא לגמרי טריוויאליים.

פתרון 2: יצירת עותק נוסף של כתובת החזרה בתחתית המחסנית, אחרי המשתנים הלוקאליים והשוואה של הכתובת כאשר חוזרים. פתרון כזה יעבוד, אבל אנחנו עלולים להיות חשופים לדריסה שמתחילה בשגרה מתחתינו.

פתרון 3: שינוי של כל המצביעים בתכנית להכיל את כתובת ההתחלה וכתובת הסיום של מקטע הזיכרון אליו הם מצביעים. הפתרון הזה ידרוש תקורה לא סבירה. בכל השמה של מצביע נצטרך להעתיק 3 מילים במקום אחת. בכל גישה למצביע נצטרך לבדוק את התחום אליו ניגשים. יש עם הפתרון הזה בעיות נוספות כמו עדכון מצביעים כאשר משנים את גודל המערך המקורי (למשל על ידי realloc).

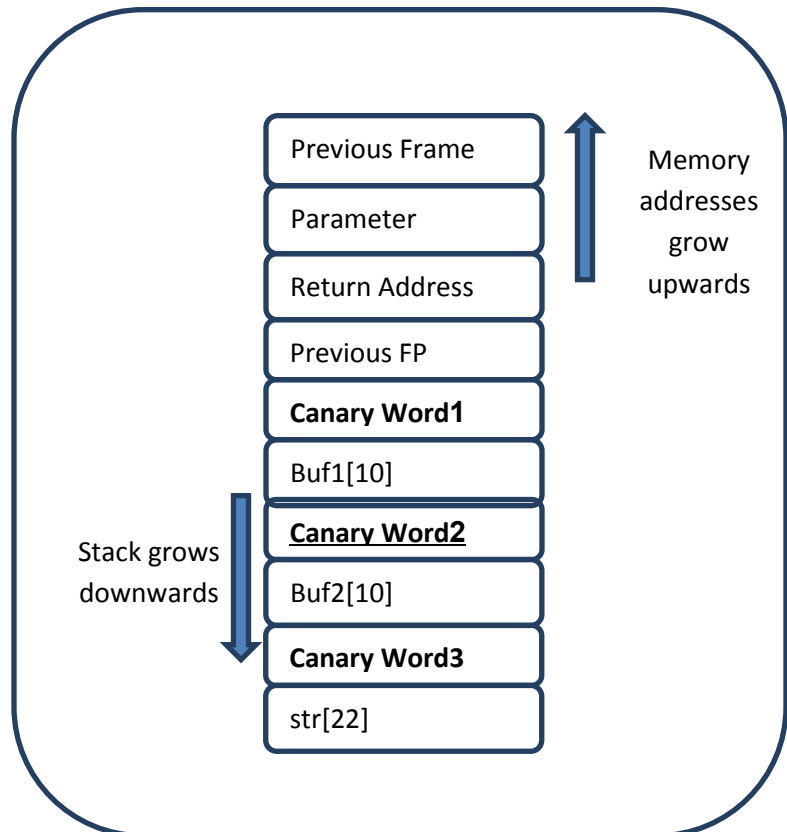
פתרון 4: ראנדומיזציה של מיקום תוכן המחסנית. הפתרון הזה יכול לעבוד. אבל התקורה שלו עלולה להיות גבוהה.

פתרון סעיף ג'

- אם מניחים שמערכים מוקצים אך ורק על המחסנית - מתי ניתן להתריע על כך שכתובה למערך אחד גולשת לתוך מערך אחר? כיצד נגלה זאת? באיזה חלקים של הקומפיילר נדרש שינוי?

ניתן להשתמש במילות קנרית בין כל שני מערכים על המחסנית. השינוי הנדרש הוא בזמן יצירת הקוד. עבור תכנית הדוגמא:

```
void bar() {
    char buf1[10];
    char buf2[10];
    char str[22];
    strcpy(str, "Lorem ipsum dolor sit");
    strcpy(buf2, str);
    char x = buf1[0];
}
```



- אם מניחים שמערכים יכולים להיות מוקצים גם באופן דינמי בזיכרון הערימה - מתי ניתן להתריע על כך שכתובה למערך אחד גולשת לתוך מערך אחר? כיצד נגלה זאת? באיזה חלקים של הקומפיילר נדרש שינוי? **שימי לב שבמקרה זה נדרשים שינויים שאינם קשורים ברשומת ההפעלה ובקריאה/חזרה משגרה.**

כעת לא נוכל יותר להשתמש בקנרית אחת עבור כל מערך. נאלץ להוסיף עבור כל מערך קנרית התחלה וקנרית סיום אשר יוספו למערך בזמן ההקצאה שלו ע"י שינוי של שגרת `malloc`. בכל כתיבה למערך נבדוק האם אנחנו דורסים מילת קנרית מיוחדת שלנו ושל מערך אחר ונדווח שגיאה.

פתרון סעיף ד'

תקורת זמן הריצה של הפתרונות בסעיף ג' היא בדרך כלל לא סבירה.

פתרון שאלה 4

א. חייב להיות שינוי במבנה העמודות על מנת לעשות שימוש בLookahead של שני תווים. אפשרות אחת היא ליצור עמודה לכל מילה בקבוצה $T \cdot (T \cup \{\$\})$ וכן עמודה עבור $\$\$$ ואז להתחיל את הניתוח מ $\$\$\$$.

ב.

דוגמא לתשובה (קיימות דרכים רבות לחשב פונקציות אלו).

הערה: STW ו- $short\text{-}Words$ הינם קיצורים של $singleTerminalWords$.

i.

נאתחל $short\text{-}Words(A) = \emptyset$ לכל $A \in V$.

- כל עוד היו שינויים באיטרציה האחרונה:

$$short\text{-}Words(A) = short\text{-}Words(A) \cup \{t \mid t \in T, A \rightarrow t\alpha \in P \wedge \alpha \rightarrow^* \varepsilon\}$$

$$short\text{-}Words(A) = short\text{-}Words(A) \cup \{t \mid A \rightarrow X\alpha \in P \wedge t \in short\text{-}words(X) \wedge \alpha \rightarrow^* \varepsilon\}$$

נכליל את ההגדרה לתבניות פסוקיות:

$$short\text{-}Words(\varepsilon) = \emptyset$$

$$\forall t \in T : short\text{-}Words(t) = \{t\}$$

$$\forall X \in (V \cup T), \alpha \in (V \cup T)^+ : short\text{-}Words(X\alpha) = \begin{cases} short\text{-}Words(X) \cup short\text{-}Words(\alpha) & \text{if } \alpha \rightarrow^* \varepsilon \wedge X \rightarrow^* \varepsilon \\ short\text{-}Words(X) & \text{if } \alpha \rightarrow^* \varepsilon \\ short\text{-}Words(\alpha) & \text{if } X \rightarrow^* \varepsilon \\ \emptyset & \text{else} \end{cases}$$

ii.

בסיס:

$$2\text{-}first(X) = \emptyset \quad \forall X \in V \cup T$$

לכל כלל $A \rightarrow X_1..X_n$ $X_i \in V \cup T$ נבצע:

$$2\text{-}first(A) = \bigcup_{i \in [n-1]} STW(X_1..X_i) \cdot first(X_{i+1}..X_n)$$

אם קיים k המקיים $X_1..X_k \rightarrow \varepsilon$, נבצע גם:

$$2\text{-}first(A) = \bigcup_{i \in [k+1]} 2\text{-}first(X_i)$$

הרחבה לתבניות פסוקיות:

ניח כי $\alpha = X\beta$, כאשר $\beta \in (V \cup T)^*$, $X \in V \cup T$:

$$2\text{-first}(\alpha) = STW(X) \cdot \text{first}(\beta) \cup 2\text{-first}(X)$$

במידה ו X אפיס נבצע גם:

$$2\text{-first}(\alpha) \cup 2\text{-first}(\beta)$$

.iii

- נאתחל $2\text{-follow}(A) = \emptyset$ לכל $A \in V$.

-לכל כלל $B \rightarrow \alpha A \beta$, נבצע(כל עוד היו שינויים באיטרציה האחרונה):

- אם $\beta \rightarrow^* \varepsilon$:

$$2\text{-follow}(A) = 2\text{-first}(\beta) \cup (\text{short-words}(\beta) \cdot \text{follow}(B)) \cup 2\text{-follow}(B)$$

- אחרת:

$$2\text{-follow}(A) = 2\text{-first}(\beta) \cup (\text{short-words}(\beta) \cdot \text{follow}(B))$$

.iv

כלל reduce $A \rightarrow \alpha$ ימצא במשבצת המתאימה למצב q ול-lookahead $t_1 t_2$ אמ"מ m בק הופיע פריט מהצורה

$$t_1 \alpha \cdot t_2 \in 2\text{-follow}(A)$$

ג.המחלקות חופפות חלקית.

- ברור כי קיימים דקדוקים בחיתוך (למשל הדקדוק "הריק" המכיל משתנה יחיד וללא כללים).
- עבור $G = (\{S\}, \{a\}, \{S \rightarrow aaa \mid aa\}, S)$ מתקיים כי $G \in SLR(2), G \notin LL(2)$.
- עבור $G = (\{S, A, B\}, \{a, b\}, \{S \rightarrow Aaa \mid Bab \mid bAab, A \rightarrow \varepsilon, B \rightarrow \varepsilon\}, S)$ מתקיים כי $G \notin SLR(2), G \in LL(2)$ (ב- $SLR(2)$ מתקבל קונפליקט במצב ההתחלתי).

ד. עבור $G = (\{S, A, B\}, \{a, b\}, \{S \rightarrow Aaa \mid Bab, A \rightarrow \varepsilon, B \rightarrow \varepsilon\}, S)$ מתקיים כי $G \notin LR(1)$ (יתקבל

קונפליקט במצב ההתחלתי בין $B \rightarrow \cdot a$ לבין $A \rightarrow \cdot a$ אך $G \in SLR(2)$, היות ו-

$$2\text{-follow}(A) \cap 2\text{-follow}(B) = \emptyset$$

העבור $G = (\{S, A, B\}, \{a, b\}, \{S \rightarrow Aaa \mid Bab \mid bAab, A \rightarrow \varepsilon, B \rightarrow \varepsilon\}, S)$ (אותו הדקדוק מסעיף ג'), מתקיים כי $G \in LR(2), G \notin SLR(2)$, קל לראות כי בLR2 הקונפליקט נפתר.

פתרון שאלה 5

Required markers:

Statement \rightarrow bidiSeq (M Exp N) { SList }

SList \rightarrow SList M Statement N ;

SList \rightarrow M Statement N ;

SList properties should include:

- 1) a sequential list of quads of the statements beginnings
- 2) a sequential list of statements' nextLists (not a merge of them !!)

E: Exp \rightarrow t

goto init

S1: S1

goto afterS1

S2: S2

goto afterS2

...

Sn: Sn

goto afterSn

init: if t>0 goto initForwards

initBackwards: t = t + 1

```
goto Sn

initForwards: t = t-1
              goto S1

//the first case: k = 1
afterS1: if t<=0 goto Exp
         t = t-1
         goto S2

//the general case: k=2 .. n-1
afterS2: if t==0 goto Exp
         if t>0 goto preS3
         t = t+1
         goto S1

preS3:  t = t-1
        goto S3

...
...
...

afterSn-1: if t==0 goto Exp
          if t>0 goto preSn
          t = t+1
          goto Sn-2

preSn:   t = t-1
         goto Sn

//the last case: k = n
afterSn: if t>=0 goto Exp
        t = t-1
        goto Sn-1
```