

23.08.2020

## מבחן סוף סמסטר – מועד ב'

### חלק ב'

#### מרצה אחראי:

ד"ר שחר יצחקי

#### מתרגלים:

אדם בוטח, שקד ברודי, רפאל כהן, יעקב סוקוליק

#### הוראות:

- א. בחלק זה של המבחן 11 עמודים מהם 7 דפי נוסחאות. בדקו שכל העמודים ברשותכם.
- ב. משך חלק זה של המבחן הוא שעה (60 דקות).
- ג. אסור כל חומר עזר חיצוני.
- ד. בחלק זה של המבחן 2 שאלות. כל השאלות הינן חובה. משקל כל שאלה מופיע בראשיתה. (חלוקת המשקל בין הסעיפים בכל שאלה אינה בהכרח אחידה).
- ה. ניתן לציין לגבי סעיף או שאלה "לא יודע/ת". תשובה זו תזכה ב- 20% מהניקוד של הסעיף או השאלה. תשובות שגויות לא יזכו בניקוד.
- ו. חובה לנמק כל תשובה. לא יינתן ניקוד על תשובות ללא נימוק.
- ז. קראו את כל המבחן לפני שאתם מתחילים לענות על השאלות.
- ח. את התשובות לשאלות יש לרשום על דפים. בתום המבחן יש לסרוק את הפתרון, להעלות אותו כקובץ PDF לאתר הקורס ולשלוח אותו לתיבת המייל של הקורס.

**בהצלחה!**

**שאלה 3 (35 נקודות): אנליזה סטטית**

בשאלה זו תידרשו ליישם ניתוח תכניות בשלב הקומפילציה. שימו לב:

- קראו את כל הסעיפים לפני שאתם מתחילים לענות.
- ניתן להשתמש בכל אנליזה סטטית שנלמדה בכיתה בציון שמה ודרך השימוש בה.
- ניתן להגדיר שורה בודדת כבלוק בסיסי (יש לציין זאת).
- יש לציין מהו ה-abstract domain ומהו יחס הסדר. יש להגדיר את הסמנטיקה האבסטרקטית המתאימה. ניתן להפנות להגדרות מחומר הקורס ואין צורך להגדירן מחדש.

נתון ייצוג ביניים אשר בו יש שני טיפוסים ערכים:  $i32$  (מילה של 32 ביט) ו- $i32^*$  (מצביע למילה של 32 ביט). האופרנדים בכל פקודה מתוויגים בטיפוסים המתאימים להם בדומה ל-LVM. כל הפעולות הן signed.

1.  $y = * (i32^*)a$  // 'y' gets the content at address 'a'
2.  $z = (i32)2 * (i32)i$
3.  $x = (i32)y + (i32)z$
4.  $*a = (i32)x$  // 'x' stored at address 'a'

בנוסף ניתן להקצות מערכים באמצעות הפקודה `alloc100`. פקודה זו תמיד מקצה מערך בגודל 100 מילים של 32 ביט כל אחת. התוצאה היא ערך מטיפוס  $i32^*$ .

1.  $a = \text{alloc100}$
2.  $\text{ofs} = (i32)4 * (i32)\text{idx}$
3.  $b = (i32^*)a + (i32)\text{ofs}$
4.  $*b = (i32) y$

א. (15 נק') תארו אנליזה סטטית שתבדוק, בכל חישוב כתובת, כלומר פעולת חיבור בין מצביע למספר, שההיסט שבו משתמשים הינו בטווח הבטוח, בהינתן המידע לעיל אודות מערכים בשפת ביניים זו.

ב. (20 נק') האנליזה מסעיף א' איננה מספיקה לצורך וידוא גישות בטוחות למערך (כלומר, היעדר buffer overflow). זאת מכיוון שניתן לחשב כתובות במספר כלשהו של צעדים.

1.  $a = \text{alloc100}$
2.  $i = (i32)220$
3.  $b = (i32^*)a + (i32)i$
4.  $c = (i32^*)b + (i32)i$
5.  $*c = (i32)y$

כתבו אנליזה נוספת, אשר מוודאת שלא משתמשים במצביע שהתקבל על ידי חיבור, כאופרנד לפעולת חיבור נוספת. כלומר, בתכנית שלעיל, האנליזה תדווח על שגיאה בשורה 4.

(שימו לב: אותה שגיאה תדווח גם אם, למשל, נשנה את הקבוע 220 ל-44 בשורה 2.)

**שאלה 4 (15 נקודות): ייצור קוד, הקצאת רגיסטרים:**  
 נתון קטע הקוד הבא בשפת FanC (אותו הקוד משאלה 1):

```

1. enum days {Sunday, Monday, Tacoday, Wednesday, Thursday,
2.             Friday, Saturday};
3.
4. int bar(enum days d2, int p, int y) {
5.     enum days z = Thursday;
6.     if ((int) d2 == (int) Tacoday) {
7.         print("It's Taco Tueeeesday!");
8.         d2 = Wednesday;
9.     }
10.    while (p < 42) {
11.        p = p + 3;
12.    }
13.    printi((int) z);
14.    return p + y;
15. }
16.
17. void main() {
18.     print("What day is it?");
19.     enum days d1 = Tacoday;
20.     printi(bar(d1, 3, 3));
21.     enum days w = Sunday;
22.     printi(bar(w, 2, 3));
23.     int t; int m; int k;
24.     k = 7;
25.     printi(bar(Monday, 2, m));
26.     m = 42;
27.     t = bar(Friday, k, 1);
28.     k = bar(w, k, t);
29.     printi((int) w);
30.     return;
31. }
```

טבלת הקצאת הרגיסטרים למשתני התוכנית:

<u>Variable</u>	<u>Register</u>
d1	r0
k	r3
m	r2
t	r1
w	r5
d2	r0
p	r3
y	r1
z	r4

הקומפיילר פועל על פי הקונבנציה שבה משתני התוכנית שמורים ברגיסטרים (ולא במחסנית) וארגומנטים לפונקציה מועברים על המחסנית.

הקומפיילר ביצע את הקצאת הרגיסטרים הנתונה בטבלה לעיל למשתני התוכנית. הציעו חלוקה של הרגיסטרים ל-caller-saved ו-callee-saved שתצמצם למינימום את מספר הכתיבות והקריאות של משתנים למחסנית (לצורך גיבוי הרגיסטרים שלהם) בסך כל הקריאות למתודה bar מתוך main.

שימו לב להנחיות בעמוד הבא, והסבירו את הצעתכם עבור כל רגיסטר ורגיסטר.

**הנחיות:**

- הניחו כי ערכים מועברים מהפונקציה הקוראת לנקראת באמצעות המחסנית בלבד.
- הניחו כי הרגיסטרים r0-r5 משמשים רק עבור ערכי המשתנים כמפורט בטבלה שלעיל, והם אינם משמשים לצורך שמירת תוצאות של חישובי ביניים.
- הניחו כי קיים רגיסטר ייעודי נפרד (שאינו מפורט בשאלה) עבור ערך ההחזרה של פונקציה.
- הניחו כי הפונקציה הנקראת (callee) בקוד שלעיל מגבה רגיסטרים שנמצאים תחת אחריותה בתחילת ריצתה.
- הניחו כי הפונקציה הקוראת (caller) בקוד שלעיל מגבה רגיסטרים שנמצאים תחת אחריותה רגע לפני קריאה לפונקציה הנקראת.
- **רמז:** היעזרו באנליזת חיות (liveness analysis).

**בהצלחה!**

## נוסחאות ואלגוריתמים

כל ההגדרות מתייחסות לדקדוק  $G = (V, T, P, S)$ .

### Top Down

$$\text{first}(\alpha) = \{t \in T \mid \alpha \Rightarrow^* t\beta \wedge \beta \in (V \cup T)^*\}$$

$$\text{follow}(A) = \{t \in T \cup \{\$\} \mid S\$ \Rightarrow^* \alpha A t \beta \wedge \alpha \in (V \cup T)^* \wedge \beta \in (V \cup T)^* (\epsilon \mid \$)\}$$

$$\text{select}(A \rightarrow \alpha) = \begin{cases} \text{first}(\alpha) \cup \text{follow}(A) & \alpha \Rightarrow^* \epsilon \\ \text{first}(\alpha) & \text{otherwise} \end{cases}$$

**הגדרה:** דקדוק  $G$  הוא  $LL(1)$  אם ורק אם לכל שני כללים ב- $G$  השייכים לאותו משתנה  $A$  מתקיים:  
 $\text{select}(A \rightarrow \alpha) \cap \text{select}(A \rightarrow \beta) = \emptyset$

הגדרת טבלת המעברים  $M: V \times (T \cup \{\$\}) \rightarrow P \cup \{\text{error}\}$  עבור דקדוק  $LL(1)$ :

$$M[A, t] = \begin{cases} A \rightarrow \alpha & t \in \text{select}(A \rightarrow \alpha) \\ \text{error} & t \notin \text{select}(A \rightarrow \alpha) \text{ for all } A \rightarrow \alpha \in P \end{cases}$$

אלגוריתם מנתח  $LL(1)$ :

```

Q.push(S)
while !Q.empty() do
    X = Q.pop()
    t = next token
    if X ∈ T then
        if X = t then MATCH
        else ERROR
    else // X ∈ V
        if M[X, t] = error then ERROR
        else PREDICT(X, t)
    end if
end while
t = next token
if t = $ then ACCEPT
else ERROR

```

**Bottom Up**

פריט LR(0) הוא  $(A \rightarrow \alpha \bullet \beta)$  כאשר  $A \rightarrow \alpha \beta \in P$   
סגור (closure) על קבוצת פריטים I מוגדר באופן אינדוקטיבי:  
 ○ בסיס:  $\text{closure}(I) = I$   
 ○ צעד: אם  $(A \rightarrow \alpha \bullet B \beta) \in \text{closure}(I)$ , אז לכל  $B \rightarrow \gamma \in P$ , גם  $(B \rightarrow \bullet \gamma) \in \text{closure}(I)$  וכלל  $B \rightarrow \gamma \in P$  גם,  $(B \rightarrow \bullet \gamma) \in \text{closure}(I)$   
פונקציית המעברים של האוטומט:  
 $\delta(I, X) = \cup \{ \text{closure}(A \rightarrow \alpha X \bullet \beta) \mid (A \rightarrow \alpha \bullet X \beta) \in I \}$

פריט LR(1) הוא  $(A \rightarrow \alpha \bullet \beta, t)$  כאשר  $A \rightarrow \alpha \beta \in P$ ,  $t \in T \cup \{ \$ \}$   
סגור (closure) על קבוצת פריטים I מוגדר באופן אינדוקטיבי:  
 ○ בסיס:  $\text{closure}(I) = I$   
 ○ צעד: אם  $(A \rightarrow \alpha \bullet B \beta, t) \in \text{closure}(I)$ , אז לכל  $B \rightarrow \gamma \in P$  ולכל  $x \in \text{first}(\beta t)$ , גם  $(B \rightarrow \bullet \gamma, x) \in \text{closure}(I)$   
פונקציית המעברים של האוטומט:  
 $\delta(I, X) = \cup \{ \text{closure}(A \rightarrow \alpha X \bullet \beta, t) \mid (A \rightarrow \alpha \bullet X \beta, t) \in I \}$

הגדרת טבלת action למנתח SLR:

$$\text{action}[i, t] = \begin{cases} \text{SHIFT}_j & \delta(I_i, t) = I_j \\ \text{REDUCE}_k & \text{rule } k \text{ is } A \rightarrow \alpha, (A \rightarrow \alpha \bullet) \in I_i \text{ and } t \in \text{follow}(A) \\ \text{ACCEPT} & (S' \rightarrow S \bullet) \in I_i \text{ and } t = \$ \\ \text{ERROR} & \text{otherwise} \end{cases}$$

הגדרת טבלת action למנתח LR(1):

$$\text{action}[i, t] = \begin{cases} \text{SHIFT}_j & \delta(I_i, t) = I_j \\ \text{REDUCE}_k & \text{rule } k \text{ is } A \rightarrow \alpha \text{ and } (A \rightarrow \alpha \bullet, t) \in I_i \\ \text{ACCEPT} & (S' \rightarrow S \bullet, \$) \in I_i \text{ and } t = \$ \\ \text{ERROR} & \text{otherwise} \end{cases}$$

הגדרת טבלת goto למנתח SLR ו-LR(1):

$$\text{goto}[i, X] = \begin{cases} j & \delta(I_i, X) = I_j \\ \text{error} & \text{otherwise} \end{cases}$$

אלגוריתם מנתח shift/reduce:

```

Q.push(θ)           // where θ is the initial state of the prefix
automaton
while true do
    k = Q.top().state
    t = next token
    do action[k , t]
end while

```

## קוד ביניים

סוגי פקודות בשפת הביניים:

1. משפטי השמה עם פעולה בינארית
  2. משפטי השמה עם פעולה אונרית
  3. משפטי העתקה
  4. קפיצה בלתי מותנה
  5. קפיצה מותנה
  6. הדפסה
- ```

x := y op z
x := op y
x := y
goto L
if x relop y goto L
print x

```

## Data-Flow Analysis

ההגדרות מתייחסות ל-CFG:  $G = (V, E)$ 

הצורה הכללית של המשוואות לסריקה קדמית:

$$\text{in}(B) = \bigcap_{(S,B) \in E} \text{out}(S) \quad \text{או} \quad \text{in}(B) = \bigcup_{(S,B) \in E} \text{out}(S)$$

$$\text{out}(B) = f_B(\text{in}(B))$$

הצורה הכללית של המשוואות לסריקה אחורית:

$$\text{in}(B) = \bigcap_{(B,D) \in E} \text{out}(D) \quad \text{או} \quad \text{in}(B) = \bigcup_{(B,D) \in E} \text{out}(D)$$

$$\text{out}(B) = f_B(\text{in}(B))$$

**ייצור קוד בשיטת Backpatching**

פונקציות:

|                                    |                                                                                                                                    |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| <code>makelist(quad)</code>        | יוצרת רשימה ריקה עם איבר אחד (ה"חור" <code>quad</code> ).                                                                          |
| <code>merge(list1, list2)</code>   | מחזירה רשימה ממוזגת של הרשימות <code>list1</code> , <code>list2</code>                                                             |
| <code>emit(code_string)</code>     | מדפיסה קוד בשפת הביניים ומאפשרת להדפיס פקודות קפיצה עם "חורים".                                                                    |
| <code>nextquad()</code>            | מחזירה את כתובת הרביעיה (הפקודה) הבאה שתצא לפלט.                                                                                   |
| <code>backpatch(list, quad)</code> | מקבלת רשימת "חורים" <code>list</code> וכתובת <code>quad</code> ו"מטילאה" את הרשימה כך שבכל החורים תופיע הכתובת <code>quad</code> . |
| <code>newtemp()</code>             | מחזירה שם של משתנה זמני חדש שאינו נמצא בשימוש בתכנית.                                                                              |

משתנים סטנדרטיים:

- `S`: גזור פקודות (statements) בשפה. תכונות:
  - `nextlist`: רשימת כתובות של פקודות המכילות חור שיש להטילא בכתובת הפקודה הבאה לביצוע אחרי הפקודה הנגזרת מ-`S`.
- `B`: גזור ביטויים בוליאניים. תכונות:
  - `truelist`: רשימת כתובות של פקודות המכילות חור שיש להטילא בכתובת אליה יש לקפוץ אם הביטוי הבוליאני מתקיים.
  - `falselist`: רשימת כתובות של פקודות המכילות חור שיש להטילא בכתובת אליה יש לקפוץ אם הביטוי הבוליאני אינו מתקיים.
- `E`: גזור ביטויים אריתמטיים. תכונות:
  - `place`: שם המשתנה הזמני לתוכו מחושב הביטוי האריתמטי.
- `M`: מרקר שמטרתו שמירת כתובת פקודה. תכונות:
  - `quad`: כתובת הפקודה הבאה מיד לאחר המרקר.
- `N`: מרקר שמטרתו דילוג על קטע קוד המוסיף קוד קפיצה שיעדה עדיין לא ידוע. תכונות:
  - `nextlist`: רשימת כתובות המכילה את מיקום פקודת הקפיצה שנכתבה ע"י `N`.



## שפת FanC

### אסימונים:

| תבנית                             | אסימון   |
|-----------------------------------|----------|
| void                              | VOID     |
| int                               | INT      |
| byte                              | BYTE     |
| b                                 | B        |
| bool                              | BOOL     |
| enum                              | ENUM     |
| and                               | AND      |
| or                                | OR       |
| not                               | NOT      |
| true                              | TRUE     |
| false                             | FALSE    |
| return                            | RETURN   |
| if                                | IF       |
| else                              | ELSE     |
| while                             | WHILE    |
| break                             | BREAK    |
| continue                          | CONTINUE |
| ;                                 | SC       |
| ,                                 | COMMA    |
| (                                 | LPAREN   |
| )                                 | RPAREN   |
| {                                 | LBRACE   |
| }                                 | RBRACE   |
| =                                 | ASSIGN   |
| ==   !=   <   >   <=   >=         | RELOP    |
| +   -   *   /                     | BINOP    |
| [a-zA-Z][a-zA-Z0-9]*              | ID       |
| 0   [1-9][0-9]*                   | NUM      |
| "([^\n\r\"\\]\ \"\\rnt\"\\ \\ )+" | STRING   |

**דקדוק:**

1.  $Program \rightarrow Enums\ Funcs$
2.  $Funcs \rightarrow \epsilon$
3.  $Funcs \rightarrow FuncDecl\ Funcs$
4.  $FuncDecl \rightarrow RetType\ ID\ LPAREN\ Formals\ RPAREN\ LBRACE\ Statements\ RBACE$
5.  $Enums \rightarrow \epsilon$
6.  $Enums \rightarrow EnumDecl\ Enums$
7.  $EnumDecl \rightarrow ENUM\ ID\ LBRACE\ EnumeratorList\ RBACE\ SC$
8.  $RetType \rightarrow Type$
9.  $RetType \rightarrow VOID$
10.  $Formals \rightarrow \epsilon$
11.  $Formals \rightarrow FormalsList$
12.  $FormalsList \rightarrow FormalDecl$
13.  $FormalsList \rightarrow FormalDecl\ COMMA\ FormalsList$
14.  $FormalDecl \rightarrow Type\ ID$
15.  $FormalDecl \rightarrow EnumType\ ID$
16.  $EnumeratorList \rightarrow Enumerator$
17.  $EnumeratorList \rightarrow EnumeratorList\ COMMA\ Enumerator$
18.  $Enumerator \rightarrow ID$
19.  $Statements \rightarrow Statement$
20.  $Statements \rightarrow Statements\ Statement$
21.  $Statement \rightarrow LBRACE\ Statements\ RBACE$
22.  $Statement \rightarrow Type\ ID\ SC$
23.  $Statement \rightarrow EnumType\ ID\ SC$
24.  $Statement \rightarrow EnumDecl$
25.  $Statement \rightarrow Type\ ID\ ASSIGN\ Exp\ SC$
26.  $Statement \rightarrow EnumType\ ID\ ASSIGN\ Exp\ SC$
27.  $Statement \rightarrow ID\ ASSIGN\ Exp\ SC$
28.  $Statement \rightarrow Call\ SC$
29.  $Statement \rightarrow RETURN\ SC$
30.  $Statement \rightarrow RETURN\ Exp\ SC$
31.  $Statement \rightarrow IF\ LPAREN\ Exp\ RPAREN\ Statement$
32.  $Statement \rightarrow IF\ LPAREN\ Exp\ RPAREN\ Statement\ ELSE\ Statement$
33.  $Statement \rightarrow WHILE\ LPAREN\ Exp\ RPAREN\ Statement$
34.  $Statement \rightarrow BREAK\ SC$
35.  $Statement \rightarrow CONTINUE\ SC$
36.  $Call \rightarrow ID\ LPAREN\ ExpList\ RPAREN$
37.  $Call \rightarrow ID\ LPAREN\ RPAREN$
38.  $ExpList \rightarrow Exp$

- 39.  $ExpList \rightarrow Exp \text{ COMMA } ExpList$
- 40.  $Type \rightarrow INT$
- 41.  $Type \rightarrow BYTE$
- 42.  $Type \rightarrow BOOL$
- 43.  $EnumType \rightarrow ENUM ID$
- 44.  $Exp \rightarrow LPAREN Exp RPAREN$
- 45.  $Exp \rightarrow Exp \text{ BINOP } Exp$
- 46.  $Exp \rightarrow ID$
- 47.  $Exp \rightarrow Call$
- 48.  $Exp \rightarrow NUM$
- 49.  $Exp \rightarrow NUM B$
- 50.  $Exp \rightarrow STRING$
- 51.  $Exp \rightarrow TRUE$
- 52.  $Exp \rightarrow FALSE$
- 53.  $Exp \rightarrow NOT Exp$
- 54.  $Exp \rightarrow Exp \text{ AND } Exp$
- 55.  $Exp \rightarrow Exp \text{ OR } Exp$
- 56.  $Exp \rightarrow Exp \text{ RELOP } Exp$
- 57.  $Exp \rightarrow LPAREN Type RPAREN Exp$