

מבוא לאופטימיזציות

תרגול 11

אופטימיזציות

- זמן
- זיכרון
- גישות לdisk
- power

תוך שימור ההתנהגות של התכנית

האופטימיזציות שנציע יהיו שמרניות

- לא נזהה את כל המקרים שניתן לשפר, אבל
- אם ביצענו שינוי כלשהו, מובטח שמשמעות התכנית לא תשתנה.

ככלל, עדיף תכנית פחות יעילה על פני תכנית לא נכונה.

אופטימיזציות

מתי לבצע אופטימיזציות?

• AST

- יתרונות: לא תלוי מכונה
- חסרונות: מעט מדי מידע מכדי להועיל
- קוד בשפת הביניים
- יתרונות: לא תלוי מכונה, חושף מספיק כדי להועיל
- חסרונות: לעיתים עדיין ניתן לבצע אופטימיזציות ספציפיות למכונה מסויימת, שלא ניתן לבצע בשלב זה.
- קוד המכונה
- יתרונות: חושף הרבה הזדמנויות לייעול
- חסרונות: תלוי מכונה, יש לממש לכל מכונה בנפרד
- במהלך הריצה (JIT)
- יתרונות: ניתן לנצל מידע הקיים רק בזמן ריצה ולא קומפילציה (פרמטרים חוזרים, פונקציות פופולריות)
- חסרונות: מתבצע על חשבון זמן ריצה

תזכורת - שפת ביניים

נעבוד עם 4 סוגי פקודות בלבד:

1. פעולה אריתמטית:
 $t_1 := t_2 \text{ binop } t_3$
2. קפיצה לא-מותנית:
goto label
3. קפיצה מותנית:
if $t_1 \text{ relop } t_2$ goto label
4. תווית:
label:

בלוק בסיסי

בלוק בסיסי הוא רצף פקודות מקסימלי כך ש:

- אינו מכיל labels, פרט לפקודה הראשונה (יעדי קפיצה)
- אינו מכיל קפיצות, פרט לפקודה האחרונה

הרעיון: בלוק בסיסי הוא רצף קוד "לינארי" – מכיל נקודת כניסה יחידה ונקודת יציאה יחידה, ולכן הביצוע בו "צפוי".

Control Flow Graph (CFG)

- גרף מכוון המייצג את זרימת בקרת התוכנית
- אחד הייצוגים הנפוצים של תכנית.
- צמתים: בלוקים בסיסיים
- קיים בלוק כניסה יחיד
- קשתות: יש קשת מבלוק A לבלוק B אם:
 - A מסתיים בקפיצה (אולי מותנית) ל-B, או
 - A מסתיים בקפיצה מותנית או ללא קפיצה ו-B מופיע אחרי A בקוד המקורי.
- כלומר קיים ביצוע (אולי) שעובר מהשורה האחרונה בבלוק A לשורה הראשונה בבלוק B.

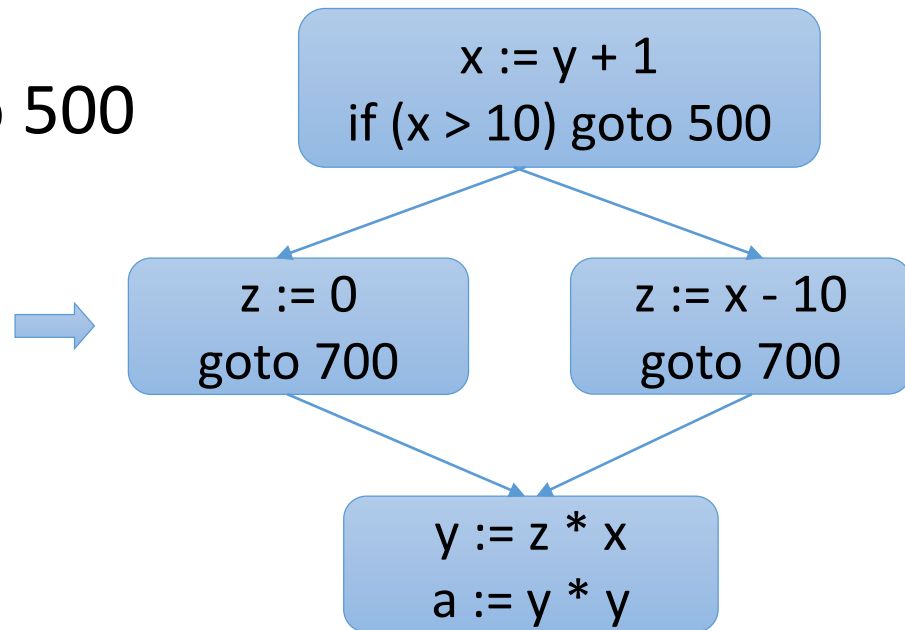
כל מתודה בתכנית תיוצג ע"י CFG כאשר הבלוק ההתחלתי יהיה הבלוק המייצג את נקודת הכניסה של המתודה

Control Flow Graph (CFG)

- לעתים נתעניין ב-CFG בו כל צומת מכיל פקודה יחידה.
- לעתים נניח כי לבלוק הראשון בתכנית אין קשתות נכנסות ולבלוק אחרון אין קשתות יוצאות
- תמיד ניתן להוסיף בלוקים ריקים בהתחלה ובסוף.

תרגיל - CFG

100: $x := y + 1$
200: if ($x > 10$) goto 500
300: $z := 0$
400: goto 700
500: $z := x - 10$
600: goto 700
700: $y := z * x$
800: $a := y * y$



איך בונים CFG?

נגדיר:

leader - כל פקודה המקיימת אחת הדרישות:

- מופיעה מייד אחרי קפיצה (מותנית או לא מותנית).
- יעד של קפיצה (מותנית או לא מותנית).
- פקודה ראשונה בתכנית.

```
100:  x := y + 1
200:  if (x > 10) goto 500
300:  z := 0
400:  goto 700
500:  z := x - 10
600:  goto 700
700:  y := z * x
800:  a := y * y
```

בדוגמה: פקודות בכתובות

700 , 500 , 300 , 100

אלגוריתם לבניית CFG

1. מצא את כל ה-leaders בתכנית.
2. עבור כל leader: הבלוק הבסיסי של ה-leader הינו:
 - סדרת הפקודות החל ממנו (כולל) ועד ל-leader הבא (לא כולל).
 - או עד לסוף התכנית.
3. הוסף קשתות בהתאם לקפיצות בסוף כל בלוק בסיסי.

שאלה ממבחן – אביב 2015 א

```
1. if (?) goto 6;  
2. c=2;  
3. d=3;  
4. d=a+1;  
5. if(?) goto 9  
6. e=3*b  
7. a=b+c;  
8. goto 6;  
9. d=b+c;  
10.if(?) goto 11  
11.e=b+c;  
12.if (?) goto 9;  
13.g=e*1  
14.f=g*e  
15.e=d+6  
16.g=e*8  
17.b=e-2  
18.g=f+d  
19.b=d+g  
20.a=e+b  
21.if (?) goto 3;  
22.goto 6;
```

נתון הקוד ביניים הבא, שהוא גוף של
פונקציה $\text{foo}(a,b,c,d,e,f,g)$

א. (3 נק') ציירו CFG לקוד הנתון
כאשר כל צומת הוא בלוק בסיסי.

שאלה ממבחן – אביב 2015 א

```
-1. if (?) goto 6;  
-2. c=2;  
-3. d=3;  
   4. d=a+1;  
   5. if(?) goto 9  
-6. e=3*b  
   7. a=b+c;  
   8. goto 6;  
-9. d=b+c;  
  10.if(?) goto 11  
-11.e=b+c;  
  12.if (?) goto 9;  
-13.g=e*1  
  14.f=g*e  
  15.e=d+6  
  16.g=e*8  
  17.b=e-2  
  18.g=f+d  
  19.b=d+g  
  20.a=e+b  
  21.if (?) goto 3;  
-22.goto 6;
```

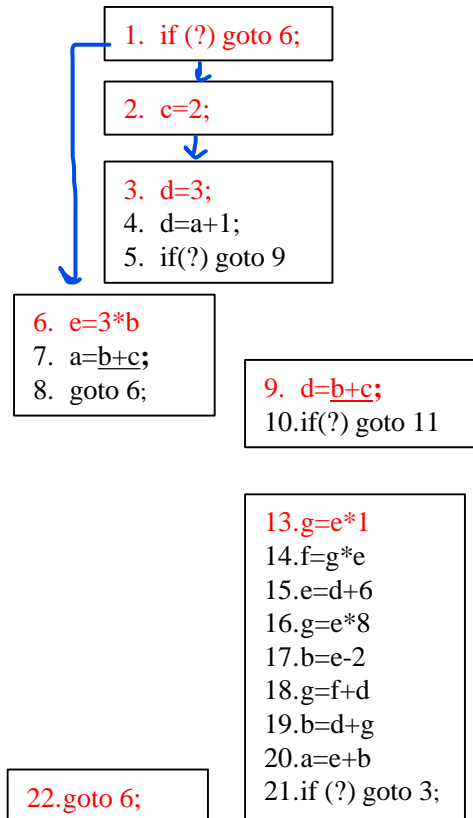
1. מצא את כל ה-leaders בתכנית.

שאלה ממבחן – אביב 2015 א

1. מצא את כל ה-leaders בתכנית.

```
1. if (?) goto 6;  
2. c=2;  
3. d=3;  
4. d=a+1;  
5. if(?) goto 9  
6. e=3*b  
7. a=b+c;  
8. goto 6;  
9. d=b+c;  
10.if(?) goto 11  
11.e=b+c;  
12.if (?) goto 9;  
13.g=e*1  
14.f=g*e  
15.e=d+6  
16.g=e*8  
17.b=e-2  
18.g=f+d  
19.b=d+g  
20.a=e+b  
21.if (?) goto 3;  
22.goto 6;
```

שאלה ממבחן – אביב 2015 א

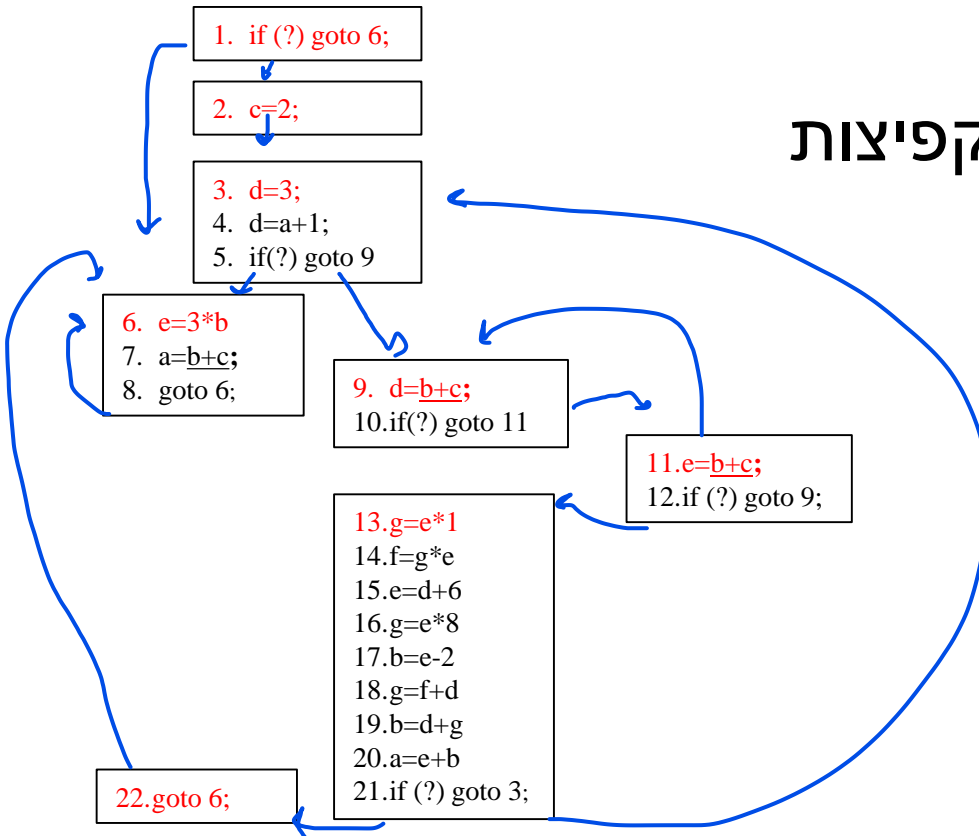


2. עבור כל leader: הבלוק הבסיסי של ה-leader הינו:

- סדרת הפקודות החל ממנו (כולל) ועד ל-leader הבא (לא כולל).
- או עד לסוף התכנית.

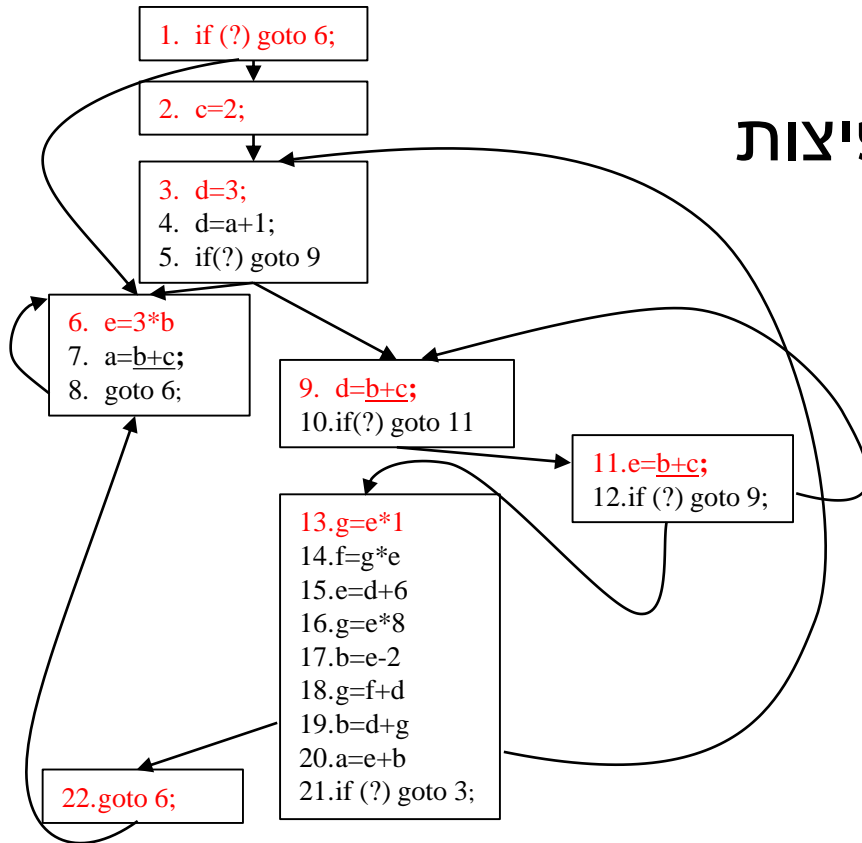
שאלה ממבחן – אביב 2015 א

3. הוסף קשתות בהתאם לקפיצות
בסוף כל בלוק בסיסי.



שאלה ממבחן – אביב 2015 א

3. הוסף קשתות בהתאם לקפיצות
בסוף כל בלוק בסיסי.



סוגי אופטימיזציות בזמן קומפילציה

• לוקאליות

מתייחסות לבלוק בסיסי יחיד. כמעט כל הקומפיילרים מבצעים אותן.

• גלובאליות

מתייחסות לגוף מתודה (ה-CFG) שלה. כלומר **אינן גלובליות בתכנית כולה, רק במתודה**. קומפיילרים רבים מבצעים אותן.

• Inter-Procedural

מתייחסות גם ליחסים בין מתודות. מסובך למימוש והתועלת נמוכה יחסית.

אופטימיזציות לוקאליות

אופטימיזציות לוקאליות

• Constant Propagation

פעפוע קבועים (בתנאי ש- a לא השתנה בין הפקודות)
מתבסס על האנליזה שראינו בשיעור 9



• Copy propagation

בתנאי ש- a, b לא משתנים בין הפקודות.
נראה מיותר, אבל בשילוב



עם אופטימיזציות אחרות יהיה
יעיל (למשל מחיקת ההשמה
($a = b$)

• מתבסס על האנליזה שראינו בתרגול 9

אופטימיזציות לוקאליות

Common sub-expression elimination •

אם x, y, z לא משתנים בין הפקודות
השינוי נאות, והורדנו חישוב
חוזר מיותר



Algebraic Simplification •

פישוטרים אלגבריים שונים
המובילים לריצה מהירה יותר או
לאופטימיזציות אחרות



• מתבסס על ידע מוקדם לגבי המעבד

• למשל, $b << 1$ יותר מהיר מ $b * 2$

אופטימיזציות לוקאליות

Constant Folding •

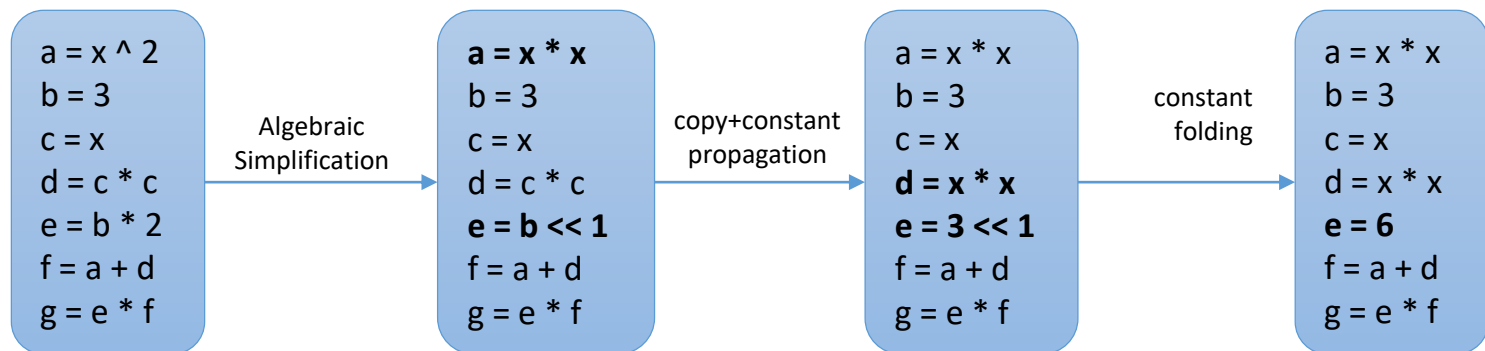
פישוט ביטויים המכילים קבועים בלבד



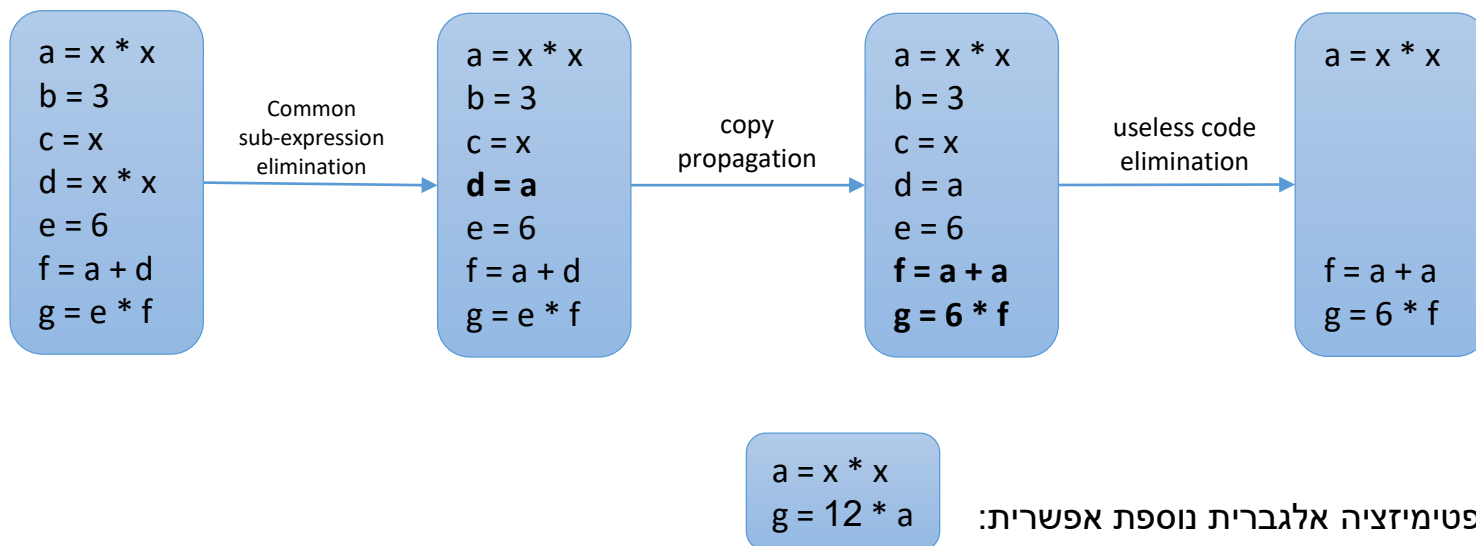
Useless code elimination •

- אם $w = \text{rhs}$ מופיע ולאחר מכן אין שימוש נוסף של w בקוד לפני השמה נוספת ל- w , ניתן להשמיט פקודה זו
- על בסיס אנליזת חיות שראינו בתרגול 9

אופטימיזציות לוקאליות - דוגמא



אופטימיזציות לוקאליות – דוגמא (המשך)



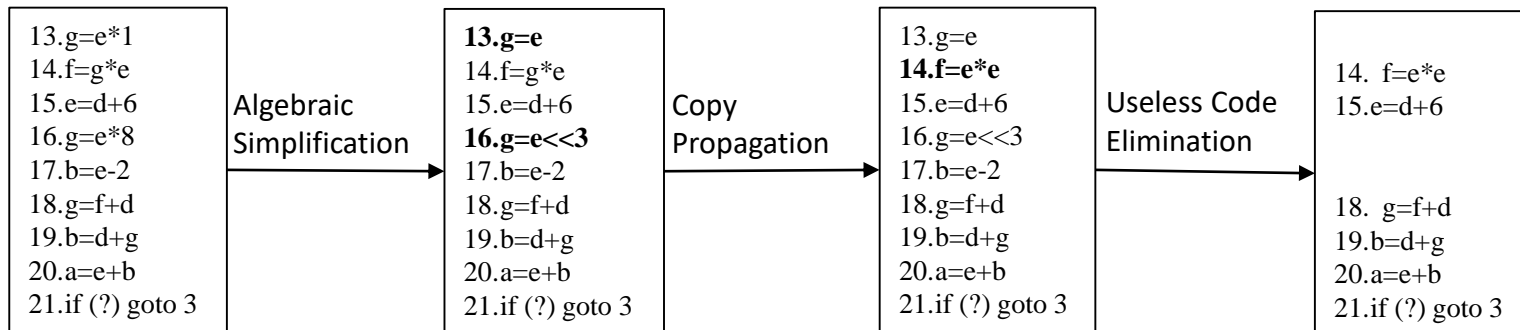
שאלה ממבחן (המשך) – אביב 2015 א

```
1. if (?) goto 6;  
2. c=2;  
3. d=3;  
4. d=a+1;  
5. if(?) goto 9  
6. e=3*b  
7. a=b+c;  
8. goto 6;  
9. d=b+c;  
10.if(?) goto 11  
11.e=b+c;  
12.if (?) goto 9;  
13.g=c*1  
14.f=g*e e=e  
15.e=d+6  
16.g=c*8 e=e+3  
17.b=e-2 d=d  
18.g=f+d  
19.b=d+g  
20.a=e+b  
21.if (?) goto 3;  
22.goto 6;
```

ג. (7 נק') עבור הבלוק שמכיל את שורה 13 בצעו מספר פעמים גדול ככל שתוכלו את פעולות האופטימיזציות:

- הפצת העתקות (Copy propagation)
 - פישוטים אלגבריים (Algebraic Simplification) על פקודה בודדת.
 - ביטול קוד ללא שימוש (Useless Code Elimination)
- עבור כל פעולת אופטימיזציה רשמו במפורש דוגמה אחת לפחות לשימוש בה.
ורשמו את הקוד המתקבל בסוף הפעלת כל האופטימיזציות שצויינו עבור הבלוק שמכיל את שורה 13.

שאלה ממבחן (המשך) – אביב 2015 א



אנליזות לצורך אופטימיזציות

- חלק מהאופטימיזציות שראינו מתבססות על אנליזות DFA (לאו דווקא gen/kill)

- מה הדומיין הרלוונטי לכל אחת מהאופטימיזציות?

- Copy propagation

$$N = \{(x, y) | x \text{ is a var, } y \text{ is a var}\}$$

- Constant propagation

$$N = \{(x, y) | x \text{ is a var, } y \text{ is a number}\}$$

- Common sub-expression elimination

$$N = \{(x, y) | x \text{ is a var, } y \text{ is an expression}\}$$

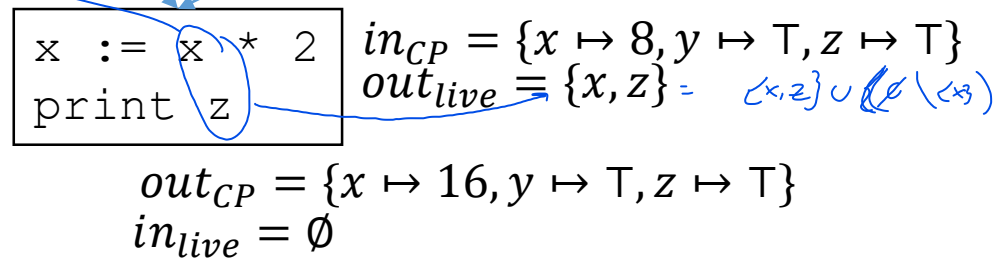
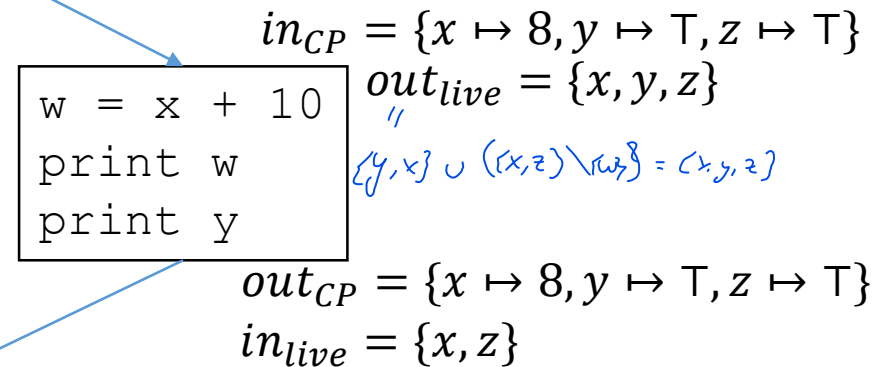
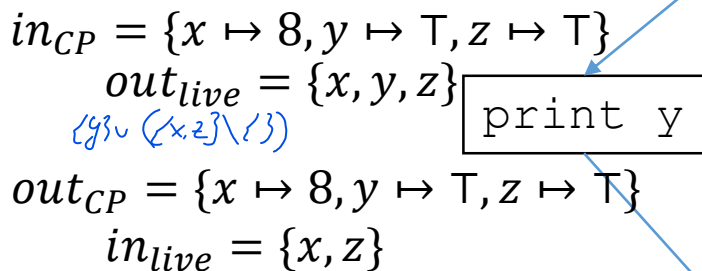
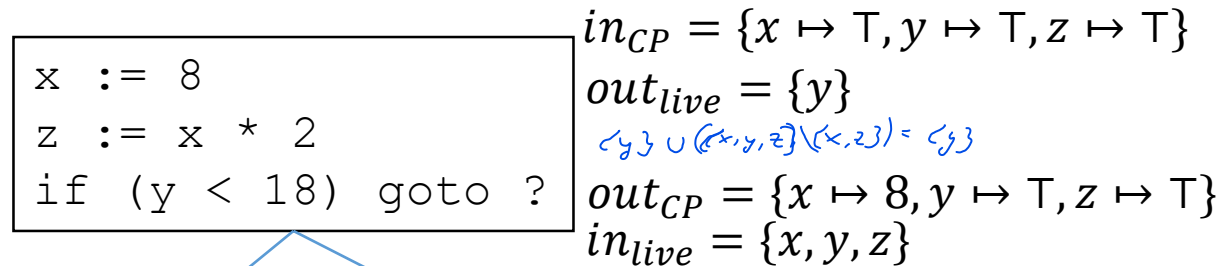
- 3 האנליזות הללו הן בעיות must עם זרימה קדמית

- וגם בעיית may עם זרימה אחורית:

- Liveness (ניתוח חיות)

$$N = \{x | x \text{ is a var}\}$$

אנליזות לצורך אופטימיזציות



אופטימיזציה לוקאלית מושפעת

```
1: x := 8
2: z := x * 2
3: if (y < 18) goto ?
```

$in_{CP} = \{x \mapsto T, y \mapsto T, z \mapsto T\}$

$out_{live} = \{y\}$

$out_{CP} = \{x \mapsto 8, y \mapsto T, z \mapsto T\}$

$in_{live} = \{x, y, z\}$

- ללא מידע מאנליזות:

- נמחק את שורה 1

- נמחק את שורה 2

- האם מותר לנו למחוק את שורה 2?

- נרצה לבצע לוקאלית

- Constant Propagation

- Constant Folding

- Algebraic Simplification

- Useless code elimination

אופטימיזציה לוקאלית מושפעת

$$\begin{array}{l} in_{CP} = \{x \mapsto 8, y \mapsto T, z \mapsto T\} \\ out_{live} = \{x, y, z\} \end{array}$$

```
1: w := 18
2: print 18
3: print y
```

$$\begin{array}{l} out_{CP} = \{x \mapsto 8, y \mapsto T, z \mapsto T\} \\ in_{live} = \{x, z\} \end{array}$$

- נרצה לבצע לוקאלית
 - Constant Propagation
 - Constant folding
 - Algebraic Simplification
 - Useless code elimination
- ללא מידע מאנליזות:
 - לא ניתן לבצע CF+CP על שורה 1
 - לא ניתן לבצע CP על שורה 2
- האם ניתן לבצע אותם?

אופטימיזציה לוקאלית מושפעת

$x := x * 2$ $\text{print } z$	$in_{CP} = \{x \mapsto 8, y \mapsto \top, z \mapsto \top\}$ $out_{live} = \{x, z\}$
-----------------------------------	--

$out_{CP} = \{x \mapsto 16, y \mapsto \top, z \mapsto \top\}$
 $in_{live} = \emptyset$

• ללא מידע מאנליזות:

Algebraic simplification •

$x := x \ll 1$ $\text{print } z$

• בהתחשבות באנליזות:

$\text{print } z$

• למה?

• נרצה לבצע לוקאלית

Constant Propagation •

Constant Folding •

Algebraic Simplification •

Useless code •

elimination

בשבוע הבא

- עוד אופטימיזציות!