את"ם – תרגיל בית מס' 4
סמסטר אביב תשע"ח

תאריך פרסום: 7/6/2018 תאריך הגשה: 28/6/2018 (בשעה 23:59)
שאלות נא לשלוח לכתובת cs234118@gmail.com. על הכותרת להכיל את המילה "HW4"

- **ההגשה בזוגות בלבד לתא ההגשה של הקורס ובאמצעות הגשה אלקטרונית.**
- **הגשות באיחור יש לתאם עם המתרגל האחראי של הקורס (לא של התרגיל) לפני מועד ההגשה הכללי.**
- **אין להגיש לתא הקורס לאחר מועד ההגשה.**

נושא התרגיל: פסיקות

בתרגיל זה שני חלקים:

- חלק א' מכיל שתי שאלות, עליהן עליכם לענות בכתב ולהגיש לתא הקורס (יש להדפיס את טופס התרגיל ולענות על גביו). אין צורך להדפיס את שאר התרגיל, רק את החלק היבש.
- חלק ב' דורש כתיבת קוד בשפת האסמבלי של PDP-11, כפי שנלמדה בהרצאות ובתרגולים. את הקוד יש לכתוב בקובץ ex4.s11, עם תיעוד פנימי ולהגיש אלקטרונית. הוראות הגשה מפורטות נמצאות **בסוף התרגיל**.

חלק א' - יבש

לשאלה זו שלושה חלקים.

חלק 1

נתון קובץ המקור הבא, המכיל פונקציה proc ושתי תוכניות prog_A ו- prog_B. ענו על הסעיפים הבאים.

```
10.          .       =       torg+20
11.          .word   proc
12.          .word   340

20.          .       =       torg+3000
21.  proc:
22.          clr     –(sp)
23.          mov     10(sp),  r0
24.          bmi     fin
25.          add     6(sp),   r0
26.  loop:   bisb    –(r0),   (sp)
27.          cmp     6(sp),   r0
28.          blo     loop
29.  fin:    mov     (sp)+,   r0
30.          rti
```

```
40.          .       =       torg+4000
41.          N       =       6
42.  prog_A:
43.          mov     #prog_A, sp
44.          mov     #N,      –(sp)
45.          mov     #a,      –(sp)
46.          iot
47.          cmp     (sp)+,   (sp)+
48.          mov     r0,      r1
49.          halt
50.  a:      .byte   1, 2, 3, 4, 5, 6

60.          .       =       torg+5000
61.          L       =       6
62.  prog_B:
63.          mov     #prog_B, sp
64.          mov     #L,      –(sp)
65.          mov     #b,      –(sp)
66.          iot
67.          add     #4,      sp
68.          mov     r0,      r2
69.          mov     r0,      r1
70.          halt
71.  b:      .byte   2, 2, 4, 4, 6, 6
```

1. מריצים את התוכנית prog_A (כלומר ערכו ההתחלתי של ה- pc הוא 4000_8). רישמו (בבסיס אוקטאלי) את ערכו של האוגר r1 עם סיום ריצת התוכנית.

r1: [ ]

2. מריצים את התוכנית prog_B (ערך pc התחלתי 5000_8). רישמו (בבסיס אוקטאלי) את ערכו של האוגר r1 עם סיום ריצת התוכנית.

r1: [ ]

חלק ב

מעוניינים לממש מנגנון המאפשר להריץ את שתי התוכניות prog_A ו- prog_B "במקביל", כששתיהן קוראות לאותו עותק של הפונקציה proc. לצורך זה מחליפים את השורות 12–10 בקוד שבחלק א בשורות 199–100 הבאות, המכילות תוכנית מעטפת main, שיגרת מלכודת של ה-T-bit, ושטחים לשימוש השיגרה. ענו על הסעיפים בעמודים הבאים.

<u>הדרכה:</u> במענה לסעיפים 3, 4 ו-5 ניתן להתעלם מהשורות המוּכָּהות.

```
100.          .       =       torg+14
101.          .word   trace
102.          .word   340
103.          .word   proc
104.          .word   360

110.          .       =       torg+2000
111.  main:
112.          mov     RegB+14,  sp
113.          mov     RegB+16,  –(sp)
114.          mov     RegB+20,  –(sp)
115.          mov     @RegA+20, Opcode
116.          rti

120.  trace:
121.          mov     r0,       @Addr
122.          mov     Addr,     r0
123.          add     #22,      r0
124.          mov     (sp)+,    –(r0)
      ;
131.          tst     Opcode
132.          bne     nohalt
133.          clr     Switch
134.          br      cont
135.  nohalt:
136.          mov     @0(r0),   Opcode
137.          mov     (sp)+,    –(r0)
138.          mov     sp,       –(r0)
139.          mov     r5,       –(r0)
140.          mov     r4,       –(r0)
141.          mov     r3,       –(r0)
142.          mov     r2,       –(r0)
143.          mov     r1,       –(r0)
```

```
151.          add     Switch,   Addr
152.          mov     Addr,     r0
153.          tst     (r0)+
      ;
161.          mov     (r0)+,    r1
162.          mov     (r0)+,    r2
163.          mov     (r0)+,    r3
164.          mov     (r0)+,    r4
165.          mov     (r0)+,    r5
166.          mov     (r0)+,    sp
167.          mov     (r0)+,    –(sp)
168.  cont:
169.          mov     (r0)+,    –(sp)
170.          mov     @Addr,    r0
171.          neg     Switch
      ;
181.          bmi     end
182.          inc     Count
183.          cmp     @RegA+20, @RegB+20
184.          bne     end
185.          inc     Match
186.  end:
187.          rtt

191.  RegA:   .word   0, 0, 0, 0, 0, 0, main
192.          .word   20, prog_A
193.  RegB:   .word   0, 0, 0, 0, 0, 0, main
194.          .word   20, prog_B
195.  Addr:   .word   RegB
196.  Opcode: .blkw   1
197   Switch: .word   RegA–RegB
198.  Count:  .word   0
199.  Match:  .word   0
```

3. עבור כל אחד מהמשפטים הבאים, סמנו אם הוא נכון או אינו נכון (על-ידי השחרת העיגול המתאים):

		נכון	**לא נכון**
א.	המנגנון מריץ פקודה מ- prog_A, לאחר מכן פקודה מ- prog_B, וחוזר חלילה, כל עוד אף אחת משתי התוכניות לא הגיעה לפקודת halt.	O	O
ב.	המנגנון עוצר כאשר אחת התוכניות (prog_A או prog_B) עומדת לבצע halt.	O	O
ג.	לכל אחת משתי התוכניות יש מחסנית משלה.	O	O
ד.	לפני שהמנגנון עובר לבצע פקודה מהתוכנית האחרת, נשמרים כל האוגרים של התוכנית הנוכחית במחסנית.	O	O
ה.	בעת הרצת פקודות של הפונקציה proc, המנגנון אינו מעביר שליטה לתוכנית האחרת עד לסיום ריצת הפונקציה.	O	O
ו.	הערכים האפשריים של המילה בכתובת Addr הם הכתובות של RegA ושל RegB.	O	O

4. רישמו (בבסיס אוקטאלי) את תוכן שש המילים הבאות מיד לפני ביצוע הפקודה iot בשורה 66 (בתוכנית prog_B שבחלק א). ניתן גם לכתוב ביטוי מהצורה "הכתובת של שורה 85" וניתן להשתמש בתוויות.

תוכן	כתובת
	RegB+14
	RegB+16
	RegB+20

תוכן	כתובת
	RegA+14
	RegA+16
	RegA+20

5. בהנחה שאוגרים r0–r5 מאותחלים ל-0 עם תחילת ריצת main, רישמו (בבסיס אוקטאלי) את תוכן שש המילים הבאות עם סיום ריצת המעבד.

תוכן	כתובת
	RegB
	RegB+2
	RegB+4

תוכן	כתובת
	RegA
	RegA+2
	RegA+4

6. משנים את שורה 104 ל:

.word 340

ומריצים את המנגנון על התוכניות prog_A ו- prog_B שבחלק א. רישמו בבסיס דצימאלי את תוכנם של המילים Count ו- Match עם עצירת המעבד.

Count: [ ] Match: [ ]

7. כיצד הייתה משתנה תשובתכם לסעיף הקודם לולא שונתה שורה 104? סמנו תשובה אחת נכונה.
   א. התשובה הייתה נותרת ללא שינוי.
   ב. הערכים ב- Count וב- Match היו גדלים פי 2.
   ג. הערך ב- Count היה גדול יותר מ-20_{10} והערך ב- Match היה נותר ללא שינוי.
   ד. הערכים ב- Count וב- Match היו שווים זה לזה וגדולים יותר מ-20_{10}.
   ה. הערך ב- Count היה גדול יותר מ-20_{10} וההפרש בינו לבין הערך ב- Match היה נותר ללא שינוי.
   ו. אף אחת מהתשובות א – ה אינה נכונה.

חלק ג

מחליפים את הפונקציה proc שבשורות 30–20 שבחלק א בפונקציה הבאה. ענו על הסעיפים שבעמוד הבא.

```
201.        tks   =      177560
202.        tkb   =      177562
203.        tps   =      177564
204.        tpb   =      177566
            .     =      torg+3000
211. proc:
212.        bisb  #1,    @#tks
213.        tstb  @#tks
214.        bpl   .–4
215.        movb  @#tkb, r0
216.        tstb  @#tps
217.        bpl   .–4
218.        movb  r0,    @#tpb
219.        rti
```

8. משנים את שורה 104 ל:

.word 340

ומריצים את המנגנון שבחלק ב על התוכניות prog_A ו- prog_B שבחלק א (עם הפונקציה proc החדשה). לאחר תחילת ריצת main, מקלידים את המחרוזת הבאה (משמאל לימין), עד לעצירת המעבד:

abcdefg…

קצב ההקלדה איטי פי אלף ויותר מזמן תגובת הבקרים וממשך מחזור פקודה. רישמו (משמאל לימין) את המחרוזת שתודפס:

[ ]

9. הניחו את התנאים של הסעיף הקודם, אך <u>ללא</u> השינוי בשורה 104. עבור כל אחד מהמשפטים הבאים, סמנו אם הוא נכון או אינו נכון (על-ידי השחרת העיגול המתאים):

	נכון	**לא נכון**
א. המחרוזת שתודפס תהיה זהה לזו שהודפסה בסעיף הקודם.	O	O
ב. התו הראשון שהוקלד ("a") יישלח להדפסה.	O	O
ג. התו השני שהוקלד ("b") יישלח להדפסה.	O	O
ד. עלול להיקרא תו מה- tkb כאשר סיבית ה- done ב- tks כבויה.	O	O
ה. רק תוכנית אחת מבין השתיים (prog_A או prog_B) תשלח תווים להדפסה.	O	O
ו. עלול להישלח תו להדפסה כאשר המדפסת אינה פנויה.	O	O

חלק ב' – רטוב (דמקה)

תיאור המשימה

בהמשך לתרגילי בית 2 ו-3, נרצה לבנות תוכנית שתאפשר לשחקנים אנושיים וממוחשבים לשחק אחד נגד השני.
לכל שחקן נגדיר timer שיגביל את הזמן שמוקצב לו לביצוע כל מהלך.
השחקן הלבן תמיד משחק ראשון.

כאשר לשחקן אנושי נגמר הזמן – המשחק נגמר

שחקן ממוחשב צריך לחשב את המהלך הכי טוב בשיטת iterative deepening (שתפורט בהמשך).

<u>תזכורת לתיאור כללי המשחק</u>

כללי המשחק הם ככללי המשחק המקורי אך מופשטים יותר.

- לוח המשחק מורכב מ- 64 משבצות (8 X 8).
- לכל שחקן יש 12 אבנים, אחד עם אבנים בצבע שחור ואחד עם אבנים בצבע לבן.
- האבנים מונחות על המשבצות השחורות של הלוח וההתקדמות היא רק על משבצות אלה באלכסון.
- בתחילת המשחק אבני השחקן הראשון מונחות בשלוש השורות הראשונות של הלוח ואילו אבני השחקן השני מונחות באותו אופן בצד שלו.
- הלבן מבצע את המהלך הראשון, והשחור משיב עליו.
- **תנועה** : כל שחקן מניע בתורו אבן-משחק באלכסון, ממשבצת שחורה אחת למשבצת שחורה סמוכה בכיוון היריב. על המשבצת להיות פנויה מכלים, כלומר לכל אבן יש שתי אפשרויות תנועה – לכיוון היריב באלכסון ימינה ולכיוון היריב באלכסון שמאלה – וכל אחת מהן עשויה להיות חסומה.
- **אכילה (דילוג)** : מתבצע כאשר אבן משחק מונחת במשבצת סמוכה לאבן היריב, ומעבר לאבן היריב יש מקום פנוי.
- אין דילוגים מרובים, כלומר כאשר מבצעים דילוג נגמר התור ואי אפשר להמשיך.
- כאשר יש דילוג אפשרי, לא חובה לבצע אותו.
- אין מלכים במשחק, כלומר כשאבן משחק מגיעה לשורה האחרונה (שורה ראשונה של היריב) היא לא הופכת למלך אלא פשוט אי אפשר להזיז את האבן הזו לאורך שאר המשחק.
- המשחק יכול להסתיים בניצחון או בתיקו. ניצחון מושג אם מתקיים אחד מהבאים :
  - לשחקן היריב לא נותרו כלל כלים על הלוח.
  - לשחקנים אין אפשרות לבצע מהלך מאחר שכליהם חסומים ולירב יש כמות קטנה יותר של כלים על הלוח.
- המקרה היחיד שמוגדר כתיקו הוא כאשר לשחקנים אין אפשרות לבצע מהלך ומספר הכלים של שני השחקנים על הלוח שווה.

הסבר טכני על Iterative Deepening

מאחר והמחשב מוגבל בזמן הריצה ולא בעומק הרקורסיה – הוא יחשב את המהלך הכי טוב באופן הבא :

1. הגדר עומק רקורסיה = 1
2. כל עוד לא נגמר הזמן :
   a. חשב את המהלך הכי טוב לפי עומק הרקורסיה שהוגדרה
   b. אם נגמר הזמן במהלך החישוב – צא מהלולאה
   c. אחרת – שמור את המהלך הכי טוב שחישבת, והגדל את עומק הרקורסיה ב-2
3. החזר את המהלך הכי טוב ששמרת

כלומר המחשב יחשב את המהלך הכי טוב בהסתכלות בעומק 1, ואז בעומק 3 וכן הלאה.

כאשר השעון יזהה שנגמר הזמן – עליו לסמן זאת ע"י הדלקת משתנה גלובאלי.

הפונקציה הרקורסיבית צריכה לבדוק את ערך המשתנה הגלובאלי לאחר כל קריאה רקורסיבית.
לדוגמא, פונקציה המחשבת את המספר הפיבונאצ'י ה-n-י ובודקת האם נגמר לה הזמן ע"י המשתנה הגלובלי TIMEOUT:

```
int fibonacci (int n) {
   if (n < 2) {return 1}
  a = fibonacci (n-1);
   if (TIMEOUT == 1) { return -1; } // -1 indicates an error
   b = fibonacci (n-2);
   if (TIMEOUT == 1) { return -1; } // -1 indicates an error
   return a + b;
}
```

בצורה זו שיטת Iterative Deepening תחזור מהחישוב מיד אחרי הדלקת המשתנה הגלובאלי.

מבנה הפקודות

כל פקודה תינתן לתוכנית כמחרוזת המתחילה בשם הפקודה ולאחריה הארגומנטים של הפקודה למשל:

move 10 01

היא פקודה חוקית מסוג move בעלת שני ארגומנטים.
הארגומנטים של פקודה מופרדים זה מזה, ומשם הפקודה, ע"י **רווח אחד או יותר**. שם פקודה אינו מכיל רווחים.

בכל מצב בו נקלטת פקודה שלא יכולה להתבצע (למשל move בתור היריב) או פקודה שאינה נתמכת (למשל AAA), יש להדפיס את הודעת השגיאה הבאה:

Cannot execute "[command]" .

כאשר [command] יוחלף בשורת הפקודה שהתקבלה (כולל הארגומנטים).

אם המחשב מבצע תנועה באמצע הקלדת פקודה – יש להתעלם מכל הקלט שהוכנס (שימו לב כי המשתמש הקליד stop אך הפקודה שנקלטה במקרה זה היא "p":

```
> sto
Black Computer move took 4.9 seconds.

Board layout:
```

```
_B_B_B_B
B_B_B_B_
_B_B___B
____B___
_____W__
W_W_W___
_W_W_W_W
W_W_W_W_

$ p

Cannot execute "p" .
```

הפקודות הנתמכות

התוכנית תתמוך ב-4 פקודות:

start – פקודה זו מקבלת ארבעה ארגומנטים המתארים האם השחקנים הם אדם או מחשב (h מסמן אדם, c מסמן מחשב), וכמה שניות יש להם לכל מהלך. מאתחלת את הלוח ומתחילה את המשחק. שני הארגומנטים הראשונים מתארים את השחקן הלבן, ושני האחרונים מתארים את השחקן השחור.
למשל – הפקודה "start h 30 c 5" מבקשת להתחיל משחק חדש, בו הלבן הוא שחקן אנושי בעל 30 שניות למהלך והשחור הוא שחקן מחשב עם 5 שניות למהלך.

שימו לב כי ניתן לשחק אדם-נגד-אדם או מחשב-נגד-מחשב.

פקודה זו מותרת רק כאשר המשחק לא פעיל.

```
$ start h 30 c 5

Starting a new game

White player is Human with 30 seconds per move

Black player is Computer with 5 seconds per move
```

stop – פקודה זו אינה מקבלת ארגומנטים, מפסיקה את המשחק הנוכחי ומדפיסה את השחקן המנצח לפי ה-WP הנוכחי (או תיקו).

פקודה זו מותרת רק כאשר המשחק פעיל, והיא מסיימת את המשחק, מדפיסה את WP עבור השחקן הנוכחי ומכריזה על המנצח.

שימו לב שפקודה זו צריכה לעבוד גם כאשר זהו תור המחשב, ובמצב זה על המחשב להפסיק את ריצתו.

```
$ stop

WP for white player is 9

White player wins!
```

move – פקודה זו מקבלת שני ארגומנטים שהם משבצות על לוח המשחק, ומזיזה את כלי המשחק של השחקן הנוכחי בהתאם (רק אם השחקן הנוכחי הוא אדם). תנועה לא חוקית תציג שגיאה.

למשל הפקודה "**move 12 01**" מזיזה כלי משחק מהמשבצת השלישית בשורה השניה (12), למשבצת השניה בשורה הראשונה (01).

שימו לב שכל ארגומנט הוא שתי ספרות המתארות את השורה ואת העמודה של אותה משבצת:

00	01	02	03	04	05	06	07
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37
40	41	42	43	44	45	46	47
50	51	52	53	54	55	56	57
60	61	62	63	64	65	66	67
70	71	72	73	74	75	76	77

לאחר ביצוע הפקודה, על התוכנית להדפיס כמה זמן לקח לשחקן לבצע את המהלך:

```
[player] Human move took [T] seconds.
```

כאשר [player] הוא השחקן שביצע את המהלך, ו-[T] הוא מספר השניות שלקח לבצע את הפקודה.

time – פקודה זו לא מקבלת ארגומנטים, ומציגה את הזמן שנותר למהלך, מעוגל לעשירית השניה:

פקודה זו מותרת רק כאשר המשחק פעיל. שימו לב שפקודה זו צריכה לעבוד גם כאשר זהו תור המחשב.

```
$ time

Time left: 2.5 seconds
```

מהלך התוכנית

1. התוכנית תדפיס הודעת פתיחה לבחירתכם, אנא הכניסו את ת.ז. שלכם להודעה. לדוגמא:

```
Welcome to [ID1] and [ID2] checkers game!
```

2. במידה והמשחק פעיל והתוכנית עוד לא הדפיסה את הלוח בתור הנוכחי:
   a. התוכנית תדפיס את מצב הלוח הנוכחי.
   b. התוכנית תציין תור מי כרגע והאם הוא אדם או מחשב.
   c. אם יש מנצח – התוכנית תציין מי המנצח ותסמן את המשחק כלא פעיל.
   d. אחרת – התוכנית תאתחל את השעון עבור השחקן הנוכחי.

3. התוכנית תדפיס את המחרוזת " $" (סימן $ ואחריו רווח בודד), ותחכה לקלט משתמש.
   a. במידה והמשחק פעיל ותור המחשב – התוכנית תחשב מהלך עבור המחשב ותחכה לקלט במקביל.
   b. כאשר המחשב יבחר מהלך – התוכנית תבצע את המהלך, תדפיס למסך את ההודעה הבאה, תדפיס שורת רווח ותחזור ל-2:

```
[player] Computer move took [T] seconds.
```

כאשר [player] הוא השחקן שביצע את המהלך, ו-[T] הוא מספר השניות שלקח למחשב.

   c. במידה והמשחק פעיל ונגמן לשחקן הזמן – התוכנית תדפיס למסך הודעה, תסמן את המשחק כלא פעיל ותחזור ל-2:

```
[player] Human is out of time. Game Over.
```

כאשר [player] הוא השחקן הנוכחי.

4. לאחר שהמשתמש יכניס פקודה וילחץ על Enter, על התוכנית לבצע את הפקודה.
   a. אם מדובר בפקודה חוקית ומותרת – התוכנית תבצע את הפקודה.
   b. אחרת – התוכנית תדפיס את המחרוזת הבאה, כאשר [command] יוחלף בשם הפקודה הלא חוקית:

```
Cannot execute "[command]" .
```

5. התוכנית תדפיס שורת רווח
6. התוכנית תחזור לשלב 2

דוגמה להרצה של התוכנית

```
Welcome to [ID1] and [ID2] checkers game!

$ start h 30 c 5

Starting a new game

White player is Human with 30 seconds per move

Black player is Computer with 5 seconds per move



Board layout:

_B_B_B_B
B_B_B_B_
_B_B_B_B
________
________
W_W_W_W_
_W_W_W_W
W_W_W_W_

Current Player: White Human

$ move 56 45

White Human move took 1.9 seconds.



Board layout:

_B_B_B_B
B_B_B_B_
_B_B_B_B
________
_____W__
W_W_W___
_W_W_W_W
W_W_W_W_
```

```
Current Player: Black Computer

$ time
Time Left: 2.5 seconds



$

Black Computer move took 5.0 seconds.



Board layout:

_B_B_B_B
B_B_B_B_
_B_B___B
____B___
_____W__
W_W_W___
_W_W_W_W
W_W_W_W_

$

White Human is out of time. Game Over.



$
```

קבלת קלט מהמשתמש

כאשר התוכנית מחכה לקלט מהמשתמש (שלב 3 בתיאור מהלך התוכנית). על התוכנית להציג את התווים שהמשתמש מקליד (echo), ולאפשר למשתמש למחוק תווים באמצעות מקש backspace.
כאשר המשתמש לוחץ על Enter, קבלת הקלט מסתיימת, והתווים המוצגים על המסך מפורשים כפקודה שיש לבצע.

ניתן להניח שאורך הקלט המקסימלי הינו 50 תווים, וניתן להתעלם מכל תו שהוקש מעבר למגבלה זו.
שימו לב כי אין להחשיב תווים שנמחקו במסגרת 50 התווים המותרים.

המתנה פעילה (Busy wait)

כל פעולות הקלט המתבצעות במהלך התוכנית צריכות להתבצע באמצעות פסיקות, אך מותר להדפיס באמצעות המתנה פעילה. המשמעות היא שאין לבנות לולאה הבודקת את מצב הדגל Done באוגר TKS, אך ניתן לבנות לולאה כזאת עבור הדגל Ready באוגר TPS.

השעון

עליכם להיעזר בפסיקת השעון בשביל למדוד את משך הזמן שעובר ולבצע פעולות מסויימות:

- בכל תחילת תור – עליכם לאתחל את הזמן של אותו תור לפי הזמן שהוגדר לשחקן בפקודת start

- אם נגמר הזמן לשחקן אדם – עליכם להציג הודעה מתאימה ולסיים את המשחק.
- אם נגמר הזמן לשחקן מחשב – עליכם לגרום לו להפסיק את החישוב ולבצע תנועה.

הזמן של התור מתחיל להיספר לאחר הדפסת הלוח והשחקן הנוכחי, **ואין להפסיק אותו במהלך התור משום סיבה** (בפרט, אין להפסיק אותו במהלך טיפול בקלט/פלט).

יש להשתמש בתווית בשם rate המצביעה למילה המציינת כמה פסיקות שעון יוזם השעון בשנייה. התווית עצמה תתוסף לתוכנית באופן אוטומטי במהלך תהליך הבדיקה באמצעות שורה כדוגמת:

```
rate:  .word 1000.
```

בגלל אופן בניית הסימולטור, מספר פסיקות השעון בשנייה משתנה ממחשב למחשב, ועשוי אף להגיע לכמה אלפים.

רמזים\טיפים

- מסיבות היסטוריות, הדפסת שורה חדשה דורשת הדפסת שני תווים:
  - Carriage Return (CR) המיוצג ב-ASCII ע"י המספר 13 (דצימלי)
  - Line Feed (LF) המיוצג ב-ASCII ע"י המספר 10 (דצימלי)

  בסימולטור, לחיצה על Enter מחזירה את התו CR בלבד. אך כדי לרדת שורה עליכם להדפיס את שני התווים. מומלץ להדפיס CR ואז LF אך התוצאה תהיה זהה גם בהדפסה בסדר הפוך.
- המקש Backspace (BS) מיוצג ב-ASCII ע"י המספר 8 (דצימלי). "הדפסת" BS תזיז את סמן המערכת מקום אחד שמאלה ללא מחיקת התווים שכבר הודפסו. אבל הדפסת תווים נוספים תתרחש על התווים הנוכחיים. חשבו איך תוכלו להשתמש בנתונים אלה כדי למחוק תווים משורת הפקודה.
- אפשר לייצג את התווים CR, LF, BS בתוך מחרוזת (.ascii<>) ע"י הצירופים \r, \n, \b בהתאמה.
- חשוב לשים לב לקדימויות של הפסיקות השונות. שימו לב למה שיקרה אם תתרחש פסיקה תוך כדי הטיפול בפסיקה קודמת.

תהליך בדיקת נכונות התוכנית

לסימולטור מספר פקודות שלא נלמדו בשיעור. אחת הפקודות הללו היא P. שימוש אחד אפשרי בפקודה זו היא השורה:

```
Pclk_cycle=1000
```

שורה זו מגדילה את מספר פסיקות השעון בשנייה. כחלק מבדיקת התרגיל, תיבדק גם נכונות הריצה של התוכנית. השורה הנ"ל תיכתב בכל הרצות הסימולטור שישמשו לבדיקת התוכנית, לפני הרצת התוכנית. אנו ממליצים לכם לבדוק את התוכנית בתנאים זהים, כלומר לכתוב את השורה הנ"ל בכל פעם שאתם מפעילים את הסימולטור (די לכתוב שורה זו פעם אחת עבור כל הפעלה של הסימולטור).

לפני הרצת התוכנית, אנו נוסיף את התווית rate לסוף הקובץ אותו אתם מגישים, בכתובת מעל 20000_8. לכן, <u>אין</u> להשתמש בכתובות מעל 20000_8 בכתיבת התוכנית. ו<u>אין</u> להגיש קובץ המכיל את הגדרת התווית הנ"ל. אתם, כמובן, רשאים להוסיף תוויות אלו במהלך כתיבת התוכנית וניפוי השגיאות (debugging), אך, כאמור, אין להגיש את התוכנית שלכם עם הגדרת התוויות הנ"ל.
לצורך הבהרת עניין זה, יסופקו שני קבצים: ex4_test.txt ו- ex4_test.bat. הקובץ ex4_test.txt מכיל את ההגדרות של תוויות אלו, והקובץ ex4_test.bat הוא קובץ הרצה המשמש להוספת התוויות. עליכם לבצע את הפעולות הבאות לפני הגשת התרגיל:

1. יש לוודא כי שם הקובץ של התוכנית הוא ex4.s11,
2. להוריד את שני הקבצים (ex4_test.txt ו-ex4_test.bat) מהאתר לאותו המיקום בו נמצא קובץ התוכנית.
3. להריץ את הקובץ ex4_test.bat.

4. ייוצר קובץ חדש בשם ex4_temp.s11 המכיל את קוד התוכנית המקורי (מהקובץ ex4.s11) וכן את הגדרת התוויות (מהקובץ ex4_test.txt). יש לוודא כי עבור הקובץ החדש אין שגיאה בזמן תרגום וכי התוכנית מביאה לפלט הצפוי.
5. בכל אופן, יש להגיש את הקובץ ex4.s11.

שימו לב: לא יתקבלו ערעורים הקשורים בעניין הטכני הנ"ל.

הערות נוספות

1. בהדפסת זמנים – עליכם להציג את הזמן תמיד בבסיס עשרוני, מעוגל לעשירית שניה.
2. שימו לב מתי כל פקודה מותרת ומתי לא (משחק פעיל/לא פעיל, שחקן אדם/מחשב וכ'ו)
3. התוכנית צריכה לפעול נכון עבור **כל** קלט תקין.
4. התוכנית צריכה לרוץ על הסימולטור המסופק באתר הקורס.
5. שימו לב כי באתר יהיה FAQ עבור תרגיל זה אשר יעודכן באופן סדיר. **אנא בדקו תחילה אם התשובה לשאלתכם מופיעה ב-FAQ.**
6. **יש להקפיד על תיעוד פנימי וחיצוני של התוכנית.** יורדו נקודות בגין תיעוד לא מלא. קיים מסמך באתר הקורס תחת לשונית תרגילי הבית המסביר **כיצד יש לתעד**.
7. שאלות על התרגיל יש להפנות **לכתובת המייל cs234118@gmail.com** בלבד.
8. **הגשות באיחור יש לתאם לפני מועד ההגשה.**
9. הגשה לתא הקורס : דף שער (נמצא באתר הקורס) + תשובות לחלק היבש (ללא דפי ההוראות של החלק הרטוב) + תיעוד חיצוני (**לכל היותר** 5-4 עמודים).

   הגשה אלקטרונית : קובץ הקוד ex4.s11 בלבד (הכולל בתוכו גם תיעוד פנימי).
10. **ההגשה בזוגות בלבד!**

עבודה נעימה!