את"ם – תרגיל בית מס' 4 - סמסטר חורף תשע"ו

תאריך פרסום:02/01/2018 תאריך הגשה: 23/01/2018 (23:55 בלילה)

- **ההגשה בזוגות בלבד לתא ההגשה של הקורס ובאמצעות הגשה אלקטרונית.**
- **שאלות על התרגיל יש להפנות לכפיר ניר-צבי** kfirnz@cs.technion.ac.il.
- **הגשות באיחור יש לתאם עם כפיר לפני מועד ההגשה הכללי.**
- **אין להגיש לתא הקורס לאחר מועד ההגשה.**

נושא התרגיל: פסיקות

בתרגיל זה שני חלקים:

- חלק א' מכיל שני חלקים. בכל חלק קטע קוד ושאלה לגביו. עליכם לענות על כל סעיפי השאלות בכתב ולהגיש לתא הקורס (יש להדפיס את טופס התרגיל ולענות על גביו).
- חלק ב' דורש כתיבת קוד בשפת האסמבלי של PDP-11, כפי שנלמד בהרצאות ובתרגולים. את הקוד יש לכתוב בקובץ ex4.s11. כדאי לקרוא באתר הקורס ב- FAQ על רמת התיעוד הנדרשת. יש להגיש את הקובץ ex4.s11 אלקטרונית דרך האתר (יש להגיש אלקטרונית רק את הקובץ ex4.s11. אין להגישו מכווץ בתור קובץ Zip כפי שמצוין באתר). **אין צורך להדפיס את הקוד. יש להגיש לתא הקורס רק את החלק היבש ואת התיעוד החיצוני של הקוד.**

חלק יבש

חלק א' עיינו בתוכנית הבאה וענו על הסעיפים שבעמודים הבאים.

```
n=4
.= torg + 14
.word    trace,    0
.= torg + 30
.word    emth,    0

.= torg + 1000
main:    mov      pc,       sp
         tst      -(sp)
         mov      #n,       r1
         emt      1
         jsr      pc,       subr
         emt      0
         halt

subr:    mov      #1,       result
         tst      r1
         beq      esubr
         dec      r1
         jsr      pc,       subr
         inc      r1
         mul      result,   r1
         mov      r1,       result
esubr:   rts      pc

.= torg + 2000
emth:    mov      (sp),     -(sp)
         sub      #2,       (sp)
         mov      @0(sp),   (sp)
         bic      #177776,  (sp)
         beq      ct
         bis      #20,      4(sp)
         br       eemth
ct:      bic      #20,      4(sp)
eemth:   tst      (sp)+
         rti

handleC: rts      pc

flag:    .word 0
         .blkw 200
spr:     .word spr
result:  .word 0
trace:   mov      @0(sp),   -(sp)
         jsr      pc,       check
         tst      (sp)+
         tst      flag
         beq      etrace
         jsr      pc,       handleC
etrace:  rtt

check:   bic      #7,       2(sp)
         cmp      2(sp),    #200
         bne      ncheck
         jsr      pc,       restore
         dec      flag
         br       echeck
ncheck:  bic      #770,     2(sp)
         cmp      2(sp),    #4000
         bne      echeck
         jsr      pc,       store
         inc      flag
echeck:  rts      pc

store:   sub      #2,       spr
         mov      r0,       @spr
         mov      r0,       -(sp)
         mov      spr,      r0
ss:      mov      r1,       -(r0)
         add      #100,     ss
         cmp      ss,       #010640
         blt      ss
         sub      #500,     ss
         mov      r0,       spr
         mov      (sp)+,    r0
         rts      pc

restore: mov      spr,      r0
rs:      mov      (r0)+,    r5
         dec      rs
         cmp      rs,       #012000
         bgt      rs
         add      #5,       rs
         mov      r0,       spr
         mov      @spr,     r0
         add      #2,       spr
         rts      pc
```

1. איזו שורה תתבצע מיד אחרי ביצוע הפקודה שבשורה 36 בפעם הראשונה?

2. רישמו, בסדר כרונולוגי משמאל לימין, את כל הערכים שמקבלת המילה בכתובת **flag** במהלך ריצת התוכנית.

---

3. מוחקים את שורה 50? סמנו את התשובה הנכונה.
   א. לא יחול כל שינוי בריצת התוכנית.
   ב. התוכנית תיכנס ללולאה אינסופית.
   ג. תהיה שגיאה בזמן ריצה.
   ד. תהיה שגיאה בזמן תרגום.
   ה. הערך הסופי ב- result אינו צפוי.
   ו. הערך הסופי ב- result ישתנה.
   ז. אף אחת מהתשובות א' – ו' אינה נכונה.

4. רישמו את תכני הכתובות המצוינות בטבלה למטה מיד **לפני** ביצוע השורה **76** **בפעם הראשונה.** ניתן לכתוב ביטוי מהצורה "הכתובת של שורה 15" וניתן להשתמש בתוויות. הניחו שבתחילת התוכנית תוכן של אוגר r_i הוא i (עבור $i<6$). אם קיים בכתובת ערך לא ידוע, כתבו "לא ידוע" במקום המתאים.

כתובת	תוכן (מספר אוקטאלי)
spr –14	
spr –12	
spr –10	
spr –6	
spr –4	
spr –2	
spr	

5. רישמו את תוכן הכתובת **spr** מיד **לפני** ביצוע השורה **79** **בפעם הראשונה.**

6. רישמו את תוכן הכתובת **result** בסוף ריצת התוכנית. (בייצוג אוקטאלי)

7. חיזרו על סעיף 6, כאשר מוחקים את שורה 11 בתוכנית. (בייצוג אוקטאלי)

8. מחליפים את שורה 83 ב- **bge rs**, רישמו את תוכן הכתובת **spr** מיד **לפני** ביצוע השורה **86** **<u>בפעם הראשונה.</u>**

<u>חלק ב'</u>

אחד הסטודנטים טען שניתן ליעל את הקוד מחלק א' ע"י שינוי מימוש השגרות הבאות:

- store
- restore
- handleC

עיינו בקוד החדש של השגרות הנ"ל בעמוד הבא, וענו על הסעיפים שאחריו.

הערה: מימוש השגרה getType אינו מופיע. הניחו שהיא ממומשת במקום אחר, כאשר:

- רגיסטר הקישור שלה הוא ה- **pc**.
- הקלט במחסנית: קידוד פקודה.
- פלט במחסנית (שמוקצה לפני הקלט במחסנית): סוג הפקודה שהתקבלה – כאשר סוג פקודה יכול להיות אחד משלושת הסוגים הבאים:
  1. פקודה שמקבלת אופרנד אחד בלבד. (**OneOp=1**)
  2. פקודה שמקבלת שני אופרנדים. (**TwoOp=2**)
  3. פקודה שמקבלת אופרנד ורגיסטר. (**RegOp=3**)
  4. **<u>אפס</u>** אחרת.
- מימושה אינו פוגע ביעילות התוכנית.

```
store:   sub   #2,        spr
         clr   @spr
         rts   pc

restore: mov   @spr,      -(sp)
         add   #2,        spr
         tst   -(sp)
loop:    mov   2(sp),     (sp)
         bic   #177770,   (sp)
         tst   (sp)
         beq   erestor
         dec   (sp)
         bis   (sp),      crest
crest:   mov   @spr,      r0
         bic   #7,        crest
         add   #2,        spr
         asr   2(sp)
         asr   2(sp)
         asr   2(sp)
         br    loop
erestor: cmp   (sp)+,     (sp)+
         rts   pc

nasl:    mov   r1,        -(sp)
         mov   6(sp),     r1
         ash   4(sp),     r1
         mov   r1,        6(sp)
         mov   (sp)+,     r1
         rts   pc
dstore:  mov   @spr,      -(sp)
         tst   -(sp)
         bic   #70,       6(sp)
         cmp   6(sp),     #5
         bgt   ignr
sds:     mov   2(sp),     (sp)
         bic   #177770,   (sp)
         beq   sss
         dec   (sp)
         cmp   (sp),      6(sp)
         beq   ignr
         mov   #-3,       (sp)
         jsr   pc,        nasl
         br    sds
sss:     jsr   pc,        auxStore
ignr:    cmp   (sp)+,     (sp)+
         rts   pc

handleC: tst   -(sp)
         mov   @4(sp),    -(sp)
         jsr   pc,        getType
         tst   2(sp)
         bne   Hcont
         br    eHandle
Hcont:   mov   (sp),      - (sp)
         bic   #177700,   (sp)
         jsr   pc,        dstore
         tst   (sp)+
         mov   #-6,       -(sp)
         jsr   pc,        nasl
         tst   (sp)+
         cmp   2(sp),     #RegOp
         beq   ord
         cmp   2(sp),     #TwoOp
         beq   tod
         br    eHandle
ord:     bic   #177770,   (sp)
         jsr   pc,        dstore
         bit   #1,        (sp)
         bne   eHandle
         inc   (sp)
         jsr   pc,        dstore
         br    eHandle
tod:     bic   #177700,   (sp)
         jsr   pc,        dstore
eHandle: cmp   (sp)+,     (sp)+
         rts   pc

auxStore: mov  @spr,      -(sp)
         mov   12(sp),    -(sp)
         mov   #6,        -(sp)
         jsr   pc,        nasl
         tst   (sp)+
         bis   (sp)+,     cmd
cmd:     mov   r0,        @spr
         bic   #700,      cmd
         mov   #3,        -(sp)
         jsr   pc,        nasl
         tst   (sp)+
         add   12(sp),    (sp)
         inc   (sp)
         sub   #2,        spr
         mov   (sp)+,     @spr
eDD:     rts   pc
```

9. כמה פעמים מגיעים לשורה 130 במהלך ריצת התוכנית? _____

10. כמה פעמים מגיעים לשורה 177 במהלך ריצת התוכנית? _____

11. מהו תפקיד התווית **flag** (תשובה מעל 10 מילים לא תתקבל)?

_____

_____

_____

12. רישמו את תכני הכתובות המצוינות בטבלה למטה מיד **לפני** ביצוע השורה **105** **בפעם הראשונה.** ניתן לכתוב ביטוי מהצורה "הכתובת של שורה 15" וניתן להשתמש בתוויות. הניחו שבתחילת התוכנית תוכן של אוגר r_i הוא i (עבור $i<6$). אם קיים בכתובת ערך לא ידוע, כתבו "לא ידוע" במקום המתאים.

כתובת	תוכן (מספר אוקטאלי)
spr –14	
spr –12	
spr –10	
spr –6	
spr –4	
spr –2	
spr	

13. מהו תפקיד השורות **143- 135**?

א. חילוץ ערכי אוגרים מתוך הפקודות.
ב. מניעת שמירה כפולה של ערכי אוגרים.
ג. שחזור אוגרים שנשמרו במחסנית.
ד. דילוג על ערכים במחסנית ששווים לערך קלט השגרה.
ה. בודקים אם נשמרו מעל חמישה אוגרים עד הנקודה הנוכחית.
ו. אף אחת מהתשובות א' – ה' אינה נכונה.

המשך השאלה בעמוד הבא

14. לפי דעתו של הסטודנט המימוש החדש הינו יעיל יותר, תארו יתרון משמעותי של המימוש החדש על המימוש הישן. (תשובה מעל 10 מילים לא תתקבל)

יתרון: _____

_____

_____

15. לפי דעתו של המרצה, יש במימוש החדש חיסרון משמעותי ביחס למימוש הישן, תארו חיסרון זה. (תשובה מעל 10 מילים לא תתקבל)

חיסרון:_____

_____

_____

חלק רטוב (מערכת להעברת תמונות מוצפנות)

תיאור המשימה

בהמשך לתרגיל בית 2 ו-3 הוחלט לבנות מערכת אינטראקטיבית להצפנה ופענוח של תמונות. מטרת המערכת היא לאפשר הצפנה של תמונות לפני שליחתן. ההצפנה תתבצע לפי האלגוריתם שתואר בתרגיל 3. המערכת תקבל פקודות בשורת הפקודה ותבצע אותן. מומלץ מאוד להשתמש בתרגיל זה בשגרות שכתבתם עבור מטלה 3.

מבנה הפקודות

כל פקודה עבור המערכת נתונה כמחרוזת המתחילה בשם הפקודה ולאחריה הארגומנטים של הפקודה. דוגמה לפקודה חוקית היא למשל:

encode 20 14

הפקודה בדוגמה זו היא מסוג encode, ויש לה שני ארגומנטים.
הארגומנטים של פקודה מופרדים זה מזה, ומשם הפקודה, ע"י **רווח אחד או יותר**. שם פקודה אינו מכיל רווחים.

הפקודות הנתמכות

המערכת תומכת ב-7 סוגי פקודות.

- encode: פקודה זו מקבלת שני ארגומנטים: הראשון מייצג את מספר השורות בתמונה והשני את מספר העמודות בתמונה. הפקודה מבקשת מהמשתמש להכניס תמונה ולאחר מכן מדפיסה את התמונה המוצפנת . דוגמה להרצת הפקודה:

```
encode 4 10
Please enter image to encode
```

בשלב זה המשתמש יכניס תמונה כפי שיוסבר בהמשך

```
The encoded image is:
```

בשלב זה תודפס התמונה המוצפנת כפי שיוסבר בהמשך

- decode: פקודה זו מקבלת שני ארגומנטים: הראשון מייצג את מספר השורות בתמונה והשני את מספר העמודות בתמונה. הפקודה מבקשת מהמשתמש להכניס תמונה מוצפנת ולאחר מכן מדפיסה את התמונה המפוענחת. דוגמה להרצת הפקודה:

```
decode 4 10
Please enter image to decode
```

בשלב זה המשתמש יכניס תמונה כפי שיוסבר בהמשך

```
The decoded image is:
```

בשלב זה תודפס התמונה המפוענחת כפי שיוסבר בהמשך

- setHash: פקודה זו מקבלת ארגומנט יחיד שהוא פונקציית הערבול אשר ישתמשו בה בפקודות האחרות. דוגמה להרצת הפקודה:

```
setHash 23456789abcdef01
Hash function set to 23456789abcdef01
```

- setKey: פקודה זו מקבלת ארגומנט יחיד שהוא מפתח ההצפנה אשר ישתמשו בו בפקודות encode ו-decode . דוגמה להרצת הפקודה:

```
setKey 01a
Encryption key set to 01a
```

- crack: פקודה זו מקבלת שלושה ארגומנטים: הראשון מייצג את מספר השורות בתמונה, השני את מספר העמודות בתמונה והשלישי מייצג אורך מפתח מקסימלי שיש לחפש. הפקודה מבקשת מהמשתמש להכניס תמונה מקורית ולאחר מכן מבקשת מהמשתמש להכניס את אותה תמונה לאחר הצפנתה. הפקודה תמצא ותדפיס את המפתח הקצר ביותר אשר מתאים עבור הקלט הנ"ל. דוגמה להרצת הפקודה:

```
crack 4 10 3
Please enter image
```

בשלב זה המשתמש יכניס תמונה כפי שיוסבר בהמשך

```
Please enter encoded image
```

בשלב זה המשתמש יכניס תמונה כפי שיוסבר בהמשך

```
The key is: [key]
```

כאשר [key] הוא המפתח שנמצא

- print: פקודה זו לא מקבלת ארגומנטים. הפקודה מדפיסה למסך את התמונה האחרונה שהודפסה ע"י הפקודות encode או decode, בגוונים של שחור לבן כך שכל פיקסל שערכו קטן מ-10 מודפס כ-' ' (רווח), וכל פיקסל שארכו גדול או שווה ל-10 מדפס כ-'*' . דוגמה להרצת הפקודה:

```
print
```

בשלב זה ,תודפס התמונה האחרונה שהודפסה על-פי הפורמט הנ"ל

- quit: פקודה זו לא מקבלת ארגומנטים. הפקודה מדפיסה למסך את המחרוזת "Goodbye!" ומסיימת את ריצת התוכנית.
-

הערות חשובות:

- הממדים של התמונה וגודל המפתח המקסימלי מועברים בבסיס אוקטלי, בעוד שהמפתח, פונקציית הערבול והתמונות מועברים בבסיס הקסדצימלי (בסיס 16), אפשר להניח שהאותיות בבסיס הקסדצימלי הן אותיות קטנות בלבד
- ניתן להניח שהפרמטרים עבור הפקודות החוקיות תקינים (מספר הפרמטרים שיתקבלו הוא כנדרש בפקודה.
- מובטח שהפקודה print תיקרא רק לאחר קריאה ל-encode או decode.
- כל שמות הפקודות הן case-sensitive, כלומר יש להקליד אותן בדיוק כמו שהן מוצגות בתרגיל (למשל אין לקבל את הפקודה PRINT, באותיות גדולות).
- הערך ההתחלתי של פונקציית הערבול היא פונקציית הזהות. (0123456789abcdef)
- הערך ההתחלתי של מפתח ההצפנה הוא 0.
- מכיוון שלא ידוע מה גודל התמונות אנחנו נקצה שלושה מרחבי זיכרון בתוויות IMG1, IMG2, IMG3. נתון שכל אחד מהם הוא בגודל של לפחות מספר הפיקסלים המקסימלי בתמונה במילים
- מובטח שמספר העמודות ומספר השורות הוא 2^7-1 לכל היותר
- הערות עבור crack:
  - ביצוע פקודה זו עלול לקחת זמן רב ולכן לאחר שמכניסים את התמונה המוצפנת ועד שמוצאים את המפתח המבוקש ניתן להכניס פקודות אחרות.
  - פקודות אשר המשתמש סיים להכניס יתבצעו לפניי המשך פיצוח המפתח.(הכוונה בסעיף זה היא שאם יש פקודה אשר מבקשת קלט נוסף, כמו encode אז יש להמשיך את חישוב crack גם בזמן שמחכים לקלט הנוסף מהמשתמש)
  - יש לדאוג שהדפסת המפתח לא תופרע ע"י פקודות אחרות.
  - אם המפתח נמצא בזמן שהמשתמש הקליד פקודה יש לדאוג שהמשתמש יוכל להמשיך מהמקום שהוא היה בו לאחר הדפסת המפתח.
  - במידה והפקודה crack לא מצאה מפתח מתאים יש להדפיס "1-" במקום המפתח
  - אפשר להניח שלא יהיה שימוש בפקודה crack ובפקודה setHash בזמן שהפקודה crack עדיין רצה.

הצפנה ופענוח של כל תמונה תתבצע לפי האלגוריתם שהוצג בתרגיל בית 3 הכולל שימוש במפתח ופונקציית הערבול. מומלץ להיעזר בקוד ובשגרות שכתבתם עבור תרגיל בית 3.

קליטת תמונה

את התמונה יש לקלוט בבסיס הקסדצימלי כאשר מתעלמים מכל תו לא הקסדצימלי, ומתעלמים מתווים אשר חורגים מגודל התמונה, למשל התמונה :

0	1	2	3
4	5	6	7
10	11	12	13
14	15	16	17

שקולה לקליטת המחרוזת :

```
0123
4567
89ab
cdef
```

וגם שקולה לקליטת המחרוזת הבאה :

```
0 12!3#45 %6*78 9a
b cdATAMefaa
```

הדפסת תמונה

את התמונה יש להדפיס בבסיס הקסדצימלי שורה אחרי שורה, למשל עבור התמונה :

1	2	1	2	0	1	2	3
3	4	3	4	4	5	6	7
5	6	5	6	10	11	12	13
7	0	7	0	14	15	16	17

תודפס המחרוזת הבאה :

```
12120123
34344567
565689ab
7070cdef
```

מהלך התוכנית

1. בתחילת ביצוע התוכנית, התוכנית תדפיס את הודעת הפתיחה

```
Welcome!
```

2. התוכנית תדפיס את המחרוזת הבאה (סימן $ ואחריו רווח בודד) :

"$ "

3. כעת התוכנית תאפשר למשתמש להכניס את הפקודה שברצונו לבצע. הפקודה יכולה להיות כל פקודה חוקית יחידה מאלו הנתמכות במערכת.
4. לאחר שהמשתמש ילחץ על המקש Enter :
   a. אם מדובר בפקודה חוקית, כלומר אחת מאלו שהוגדרו לעיל :
      i. התוכנית תבצע את הפקודה שהתקבלה על המסך.
      ii. התוכנית תדפיס למסך את המחרוזת הבאה :

```
"[command] completed in [T]sec"
```

כאשר [command] מוחלף בשם הפקודה שזה עתה התבצעה.
ו-[T] הוא מספר השניות שלקח לבצע את הפקודה מעוגל לעשיריות שניה.
יש לכתוב את הזמן בבסיס עשרוני.

   b. אם מדובר בפקודה שאינה חוקית, כלומר לא אחת מאלו שהוגדרו לעיל :
      i. התוכנית תדפיס למסך את המחרוזת הבאה :

```
"Unknown command [command]."
```

כאשר [command] מוחלף בשם הפקודה הלא חוקית.

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

דוגמה להרצה של התוכנית

```
Welcome!
$ setHash 23456789abcdef01
Hash function set to 23456789abcdef01
setHash completed in 0.1sec
$ crack 4 10 3
Please enter image
1212 0123
3434 4567
5656 89ab
7070 cdef
Please enter encoded image
3c3c@234d
2121 523c
70-70-c5-ab
6d6db4da
$ encrypt 10 40
Unknown command encrypt.
$ decode 10 40
Please enter image to encode
234a56
The key is: 04
crack completed in 64.1sec
234a5678 bcdef019 234a5678 b92345c2
```

יש לשים לב שהחלק המודגש בשורה נכתב ע"י המחשב מבלי שהיה צורך להקליד אותו בשנית (חלק זה הוקלד לפני מציאת המפתח)

```
867d3e92 f4562789 53408167 2ab923cd
564e897f 40237856 2391564a c8b7e2d9
234f6075 92184563 789a3b42 56dc8e27
f01abcd9 234e5678 f01abcd9 e23456f2
90784231 a5673892 6b45978c 4d23256e
89f70423 71562389 56a4b978 34c2d256
89274563 89274563 89274563 89274523
The decoded image is:
012834569abcdef701283456970123a0
45b6c701234d567012e3f45689701ab0
5c345d67012e34567f01283495a6b7c0
d01234e5670f123485670192a34b56c0
def89ab7012c3456def89ab7c01234d0
e56701f234586701923456a7b01234c0
d56701e2345f670182345697a01234b0
567012345670123456701234567012 30
decode completed in 209.1sec
$ print
   *    *******    *    *     *
  * *      *      * *   **   **
 *   *     *     *   *  * * * *
*     *    *    *     * *  *  *
*******    *    ******* *     *
*     *    *    *     * *     *
*     *    *    *     * *     *

print completed in 0.1sec
$ quit
Goodbye!
```

קבלת קלט מהמשתמש

בחלק זה נסביר כיצד על התוכנית להתנהג כאשר היא מחכה לקלט מהמשתמש (שלב 3 בתיאור מהלך התוכנית לעיל). על התוכנית להציג את התווים שהמשתמש מקליד (echo), ולאפשר למשתמש למחוק תווים שהקליד באמצעות שימוש במקש Backspace. כאשר המשתמש לוחץ על Enter קבלת הקלט מסתיימת, והתווים המוצגים באותה עת על המסך מפורשים כפקודה שאותה יש לבצע.
ניתן להניח שאורך הקלט המקסימאלי המתקבל הינו 50 תווים. אם המשתמש מנסה להקיש תווים נוספים, מעבר ל-50 המותרים, על התוכנית להתעלם מהם. נדגיש כי אם במהלך ההקלדה הוקש Backspace אין להחשיב את התווים שנמחקו במסגרת 50 התווים המותרים, וניתן להכניס תווים חדשים במקומם.

המתנה פעילה (Busy wait)

כל פעולות הקלט המתבצעות במהלך התוכנית צריכות להתבצע באמצעות פסיקות. לעומת זאת, מותר להדפיס באמצעות המתנה פעילה. המשמעות היא שאין לבנות לולאה הדוקה הבודקת את מצבו של הדגל Done באוגר המנשק TKS, אך ניתן לבדוק בלולאה כזו את מצבו של הדגל Ready באוגר המנשק TPS.

השעון

עליכם להיעזר בפסיקת השעון בשביל למדוד את משך הזמן שעובר. יש למדוד את כל הזמן מרגע הכנסת הפקודה ועד לסיומה, כלומר יש לכלול גם את זמן החישוב וגם את זמן ההמתנה לקלט מהמשתמש אם יש כזה.

יש להשתמש בתווית בשם rate המצביעה למילה המציינת כמה פסיקות שעון יוזם השעון בשנייה. עליכם להשתמש בנתון זה ולא להניח שהוא 50 כפי שנלמד בשיעור. 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

בחלק זה יוצגו פריטי מידע ומספר המלצות מימוש כדי לעזור לכם לבנות את התוכנית. אינכם מחויבים להמלצות המופיעות בחלק זה, ואתם יכולים לבחור דרכי מימוש אחרות אם אתם מעוניינים בכך.

- מסיבות היסטוריות המקש Enter מזוהה עם זוג תווי ה-ASCII:
  - Line Feed (LF) המיוצג על-ידי המספר 10 (בסיס דצימלי).
  - Carriage Return (CR) המיוצג על-ידי המספר 13 (בסיס דצימלי).

  בסימולטור שבידיכם כאשר המשתמש לוחץ על המקש Enter, התו שיתקבל יהיה CR בלבד. לעומת זאת, כאשר ברצונכם לעבור לשורה הבאה, עליכם להדפיס את התו LF ולאחריו את התו CR.
- המקש Backspace (BS) מיוצג בטבלת ה-ASCII על-ידי המספר 8. הדפסת התו BS תחזיר את "סמן המערכת" מקום אחד שמאלה: כלומר, אם התו הבא שיוקלד אמור להיות מודפס במקום החמישי בשורה ונקליד לפניו 3 פעמים את התו BS, אז התו יודפס במקום השני בשורה ולא במקום החמישי. נדגיש כי התווים שהודפסו כבר לא ימחקו, אבל ניתן כמובן להדפיס תווים אחרים שיחליפו אותם. חישבו כיצד ניתן להשתמש בתכונה זו בשביל למחוק תווים שהודפסו.
- אפשר לייצג את התווים LF, CR, ו-BS בתוך מחרוזות (<ascii>.) ע"י הצירופים \n, \r, ו-\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 התוויות rate, IMG1 , IMG2 ו IMG3 לסוף הקובץ אותו אתם מגישים, כל אלו בכתובות מעל 20000_8. לכן, אין להשתמש בכתובות מעל 20000_8 בכתיבת התוכנית. כמו כן, אין להגיש קובץ המכיל את הגדרות התוויות הנ"ל (שכן הגדרות אלו מוספות במהלך הבדיקה). 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_test1.txt ו- ex4_test.bat. הקובץ ex4_test1.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_test1.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_test1.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. התוכנית צריכה לפעול נכונה עבור **כל** קלט.
2. התוכנית צריכה לרוץ על הסימולטור המסופק באתר הקורס.
3. שימו לב כי באתר יהיה FAQ עבור תרגיל זה אשר יעודכן באופן סדיר. **אנא בדקו תחילה אם התשובה לשאלתכם מופיעה ב- FAQ**.
4. **יש להקפיד על תיעוד פנימי וחיצוני של התוכנית.** יורדו נקודות בגין תיעוד לא מלא. ניתן לקרוא באתר הקורס ב-FAQ על רמת התיעוד הנדרשת.
5. שאלות על התרגיל יש להפנות **לכפיר ניר-צבי** בלבד במייל kfirnz@cs.technion.ac.il.
6. **הגשות באיחור יש לתאם לפני מועד ההגשה.**
7. הגשה לתא הקורס: חלק יבש + תיעוד חיצוני (אין צורך להגיש את התוכנית מודפסת).
   הגשה אלקטרונית: קובץ הקוד ex4.s11 בלבד.

עבודה נעימה!