מערכות הפעלה

234123

2

תרגיל בית יבש מספר:

גל לילוס

שם

031175318

מספר זהות

lilos.gal@gmail.com

דואר אלקטרוני

אפרת אסולין

שם

301254561

מספר זהות

e.z.asulin@gmail.com

דואר אלקטרוני

ציון

אלכס ביבה

שם

**304370570**

מספר זהות

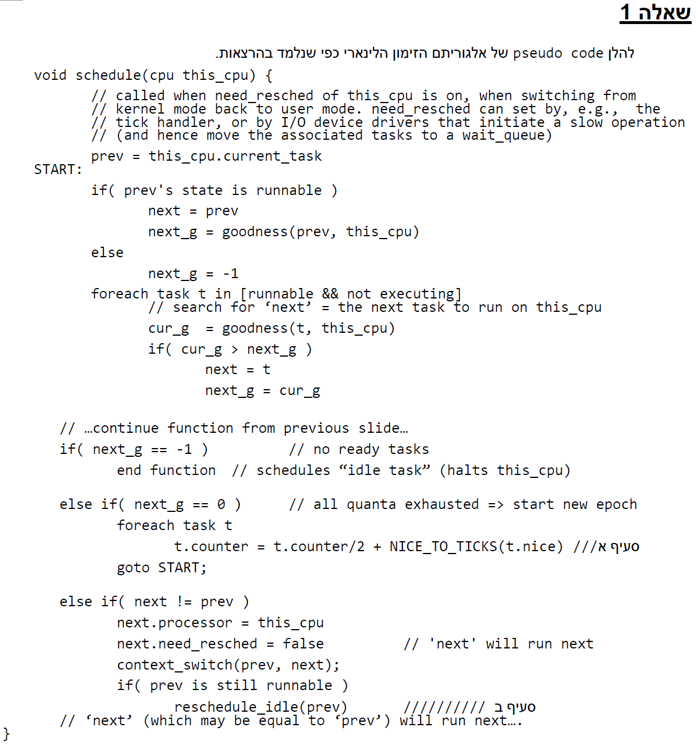
alexbiba@gmail.com

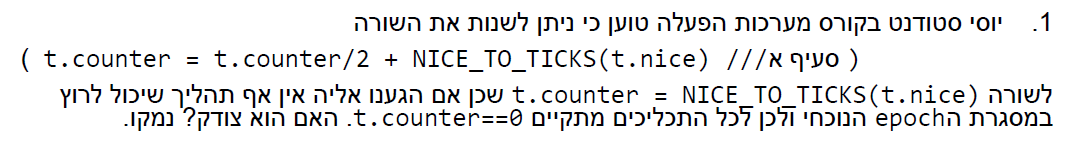
דואר אלקטרוני

להחזיר לתא מס'

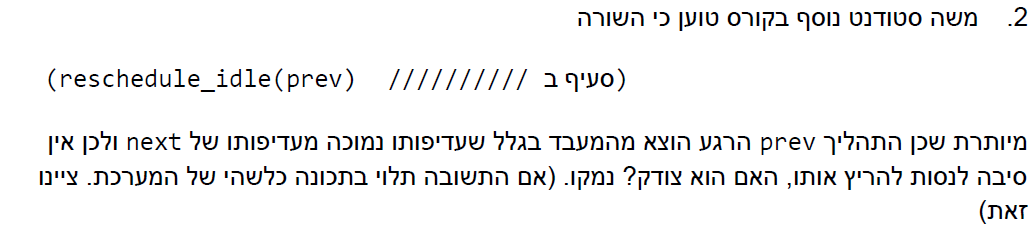
הוגש ע"י:

30

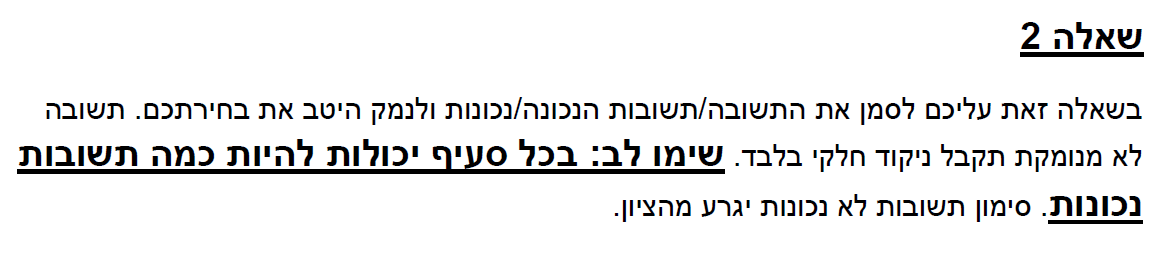


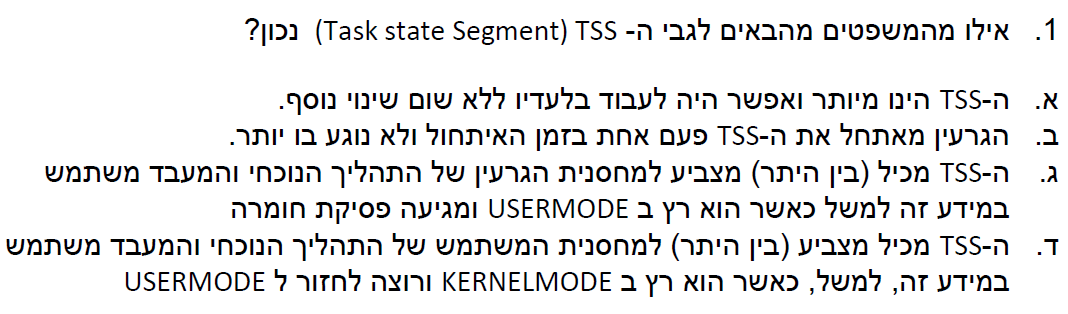


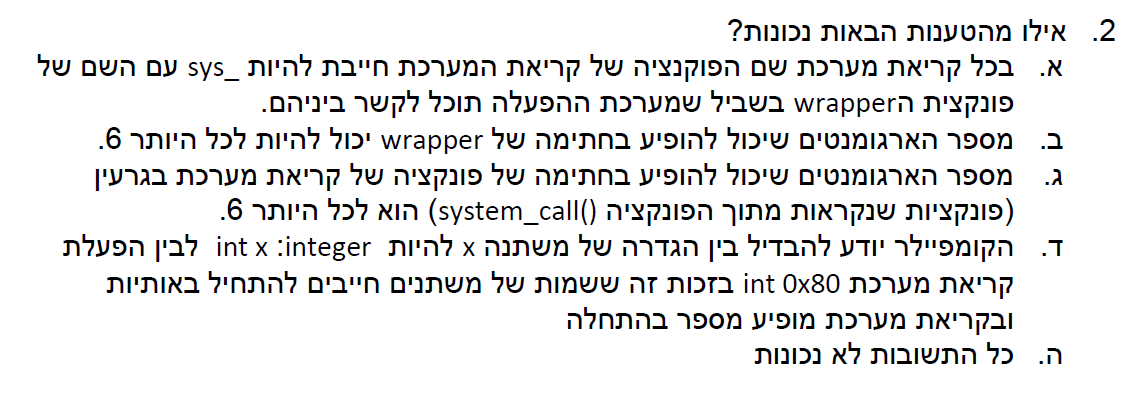
פתרון: יוסי טעה  
יש תהליכים שנמצאים בwait\_queue ומבצעים עליהם \_\_wake\_up\_common   
כאשר הוצאנו אותם ל WAIT,יתכן והcounter שלהם עוד לא התאפס לכן כשנחזיר אותם לrunnable ,  
מכיוון שבתחילת הepoch הבא הcounter החדש ישתמש בcounter הישן באופו הבא   
task.counter(new)= tesk.counter(old)+NICE\_TO\_TICKS  
לcounter הישן שאינו 0 יש חשיבות לcounter החדש  
ולכן לא ניתן לשנות את הנוסחה



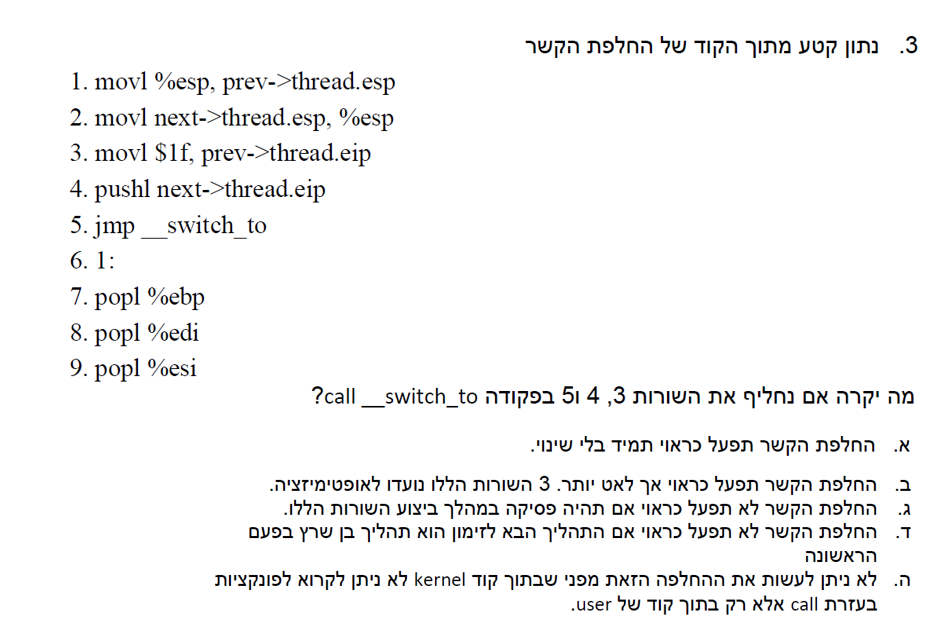
פתרון:משה טועה  
התכונה שאליה משה לא התיחס היא מספר המעבדים  
עבור מעבד אחד – משה צודק .

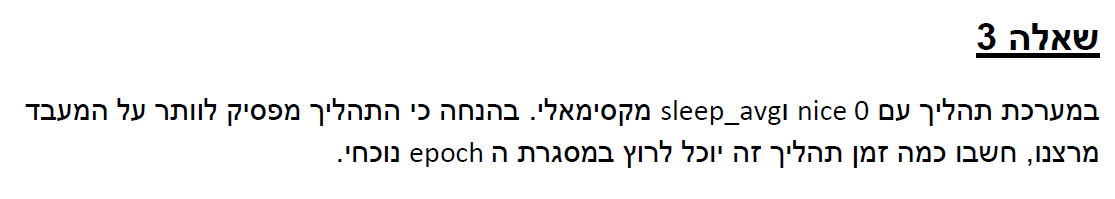


פתרון:  
א) לא נכון : כאשר עוברים מ USER MODE ל KERNEL MODE . החומרה צריכה לשמור את הנתונים esp ,eip,ss,cs,eflags במחסנית הגרעין המתאימה (לחזור מKERNEL לUSER פשוט משתמשים בנתונים אלו - iret) כדי לדעת לאיזה מחסנית בגרעין היא צריכה לגשת. חייבים מצביע למחסנית הנוכחית בKERNEL .  
בתהליך ההפוך (מעבר מKERNEL לUSER ) אנו יודעים לשחזר את הרגיסטרים כי מלכתחילה הם נשמרו במחסנית הגרעין.מנגון זה לא קיים במעבר מUSER לKERNAL כי אין נתונים שנשמרים במחסנית ה USER שאומרים לאן לגשת בKERNEL (אין int128 ו iret) ולכן חיבים מנגנון כלשהו שיבצע את ההחלפה .זה מבוצע עלידי   
esp0 - זהו השדה שמצביע לתחתית המחסנית הגרעין הנוכחית (CURRENT ) נמצא בTSS והוא מתעדכן כל החלפת הקשר  
ב)לא נכון : esp0 - השדה שמצביע לתחתית המחסנית הגרעין הנוכחית (CURRENT ) נמצא בTSS והוא מתעדכן כל החלפת הקשר בפונקציה \_switch\_to שנקראת עלידי מאקרו \_\_switch\_to שנקראת עלידי פונקצית contect\_switch (שנקראת בהכרח כאשר יש החלפת הקשר יזום או עלידי פסיקה)  
ג) **נכון** :כמתואר בסעיף א  
ד)לא נכון : בשביל זה משתמשים במנגנון של שמירת הנתונים .כשעוברים מUSER לKERNEL ושיחזורם כשחוזרים לUSER MODE כמתואר בסעיף א

 פתרון:  
א)לא נכון :הקישור נעשה דרך רגיסטר EAX שמעביר מספר שירות של הקמ"ה הרצוי (מתוך הרשימה SYS\_CALL\_TABLE ) כשאנו קוראים לSYSTEM\_CALL . כתיבת SYS\_ הוא מוסכמה (למען הסדר הטוב)  
כאשר X הוא בUSER אז SYS\_X הוא בגרעין

ב)לא נכון: הארגומנטים שמועברים לWRAPPER מועברים בשיטה הרגילה C CALLING CONVANTION (אין מעבר מUSER לKERNEL ) ולכן ניתן להעביר להעביר מספר ארגומטרים כרצוננו  
  
ג)נכון. wrapper מעביר לכל היותר 6 פרמטרים ל-system\_call ברגיסטרים ebx,ecx,edx,esi,edi,ebp בחתימה של הפונקציה לה קוראת (פונקציית קמ"ה) אם נכניס כמה יותר מ6 פרמטרים למשל   
 int my\_open(int x1,int x2,……int x100), אנו עלולים להשתמש ברגיסטרים שאינם מיועדים עבור הפונקציה.  
למשל אלו שנשמרו עלידי int 128 כמו ss,esp,eflags,cs,eip .  
  
ד)לא נכון: הקומפיילר יזהה את הפקודות לפי השפה בה הם נכתבו.לכן int 0x80 בקוד אסמבלי יתן פסיקה.ו int 128 יתן שגיעה.אם נרשום ב C רגיל int 128 לא נקבל פסיקה אלה הגדרת משתנה . ואם נרשום int 0x80 נקבל שגיעה



פתרון:  
א) לא נכון : בגלל סעיף ד  
ב) לא נכון : זה **לא** נועד לאופטימיזציה .האופימיזציה היחידה שיש היא שקוראים ל \_\_switch\_to עלידי FASTCALL  
שדואגת להעביר את הפרמטרים ברגיסטרים ולא במחסנית  
ג) לא נכון :פונקצית context\_switch בתבצעת בפסיקות חסומות כלומר דגל IF=0 (interrupt flag )ולכן לא מתאפשרות פסיקות חומרה (ככלל הגררעין של LINUX הוא non-preemptive )  
ד) **נכון** : המנגנון של שורות אלו שונה מקריאת CALL בכך שהוא מאפשר חזרה מהפונקציה ,למקום שונה מהמקום שהפונקציה הסתיימה,בפרט ,להבדיל בין החלפת הקשר לתהליך חדש "שיצא" מfork לבין תהליך שכבר עבר החלפת הקשר:  
- עבור תהליך B שכבר ביצע context\_switch בעבר . בthread.eip נשמר לייבל 1 ממנו הוא סיים בעבר ולכן  
כשיחזור מ\_\_switch \_to בהחלפת הקשר הנוכחית הוא יחזור לליבל זה והמחסנית תהיה כפי שעזבנו אותה(כמו מנגנון call ) .   
- עבור תהליך שרץ בפעם הראשונה – בthread.eip נשמר לייבל ret\_from\_fork (כשקיבל בעת יצירתו עלידיfork ) פונקציה זו דואגת להחזיר תהליך בן חדש שנוצר עלידי fork לריצה (כאשר esp ו eip יהיו כמו האב ו EAX בערך חזרה יהיה 0 ) לכן כשיחזור מ\_\_switch \_to בהחלפת הקשר הנוכחית הוא יקפוץ לפונקציה ret\_from\_fork ולמעשה נבצע החלפת הקשר לתהליך חדש שהוא עותק של האב.  
  
לכן.אם נחליף שורות אלו בקריאה סטנדרטית לפונקציה (call) עבור תהליך שרק נוצר (ועוד לא עבר החלפת הקשר) לא נגיע ל ret\_from\_fork (ולא יתבצע ret\_from\_sys\_call),לכן ריצת הבן תמשיך לא מUSER מהמקום שבו האב ביצע FORK אלה בKERNEL בהמשך התהליך שממנו הוא נקרא וזה כמובן שגיעה (זמן ריצה)  
ה)לא נכון: הקריאה לפונקציות בתוך הKERNEL היא גם עלידי C calling convertion למשל schedule קוראת רגיל ל context\_switch  
  
  
  


פתרון:

תחילה נחשב כמה time slice התהליך מקבל כל פעם שהוא יוצא מתור ruqueue active (rq->active )

עבור הנוסחה

עבור nice = 0 נקבל static\_prio = 120 ולכן

נראה אם תהליך זה יכול להיכנס שוב באפוק הנוכחי – כלומר אם הוא אינטרקטיבי ואין הרעבה של תהליכים אחרים (גם רשום בPIAZZA שניתן להניח שהוא התהליך היחיד)  
נראה שהוא אינטרקטיבי

אין הרעבה מכיוון שלא נתון לנו nr\_running ולא נתון STARVATION\_LIMIT לא ניתן לחשב אם יש הרעבה ולכן נניח שאין  
סהכ התהליך יכול להיות פעמיים בEPOKE הנוכחי כלומר time\_slice\*2 כלומר (לפי החישוב של time\_slice )

151\*2 = 302 HZ