ארגון המחשב ושפת סף – תרגיל 4

סמסטר ב', 2016

צבי מלמד

הנחיות הגשה:

- **מועד הגשה:. עד ליום ראשון 19/6/16 בחצות (חצות שבין יום ראשון לשני).**

הערה לגבי התאריך: מכיוון שאנו קרובים למועד הבחינה, אינני רוצה שתעבדו על התרגיל ממש לפני הבחינה במקום להתכונן אליה. יחד עם זאת, התרגיל עצמו מהווה חלק מההכנה לבחינה. תלמידים שלא יספיקו לסיים את התרגיל לפני הבחינה, תינתן להם האפשרות להגיש אותו במועד מאוחר יותר. במקרה כזה תהיה הורדה סמלית של לכל היותר 10 נקודות.

- **קבוצות עבודה:** יש לבצע את התרגיל לבד. אין הגשה משותפת!

- **אופן הגשה:** תיבת הגשה ב MAMA.
  - יש להגיש קבצי מקור בלבד. לכל שאלה בתרגיל יש לשלוח קובץ **asm.** בלבד של הפתרון. כלומר, אין להגיש את תוצרי הקומפילציה (למשל קבצי exe.). במקום שמצוין יש להגיש גם את קובץ הנתונים שעושים לו include מתוך התכנית.
  - שמות הקבצים צריכים להיות: ex2_q1.asm, ex2_q2.asm,.
  - את כל קבצי ההגשה יש לארוז בקובץ **ZIP.**
  - יש לרשום את השם ותעודת הזהות כהערה בקוד בראש כל קובץ ה- **asm.**
  - **יש לוודא שהתוכנית שאתם שולחים עובדת קומפילציה ומצליחה לרוץ – שאלה שלא עובדת קומפילציה, תקבל ציון 0.**

- **צילומי מסך:**
  - לכל שאלה עליכם לצרף צילום מסך של החלון שבו קימפלתם והרצתם את התרגיל. לשם כך השתמשו ב ALT+Print-screen או בכלי snipping tool של WINDOWS.
  - אין לשלוח צילום של כל המסך!
  - שמות הקבצים של צילומי המסך צריכים להיות כמו קבצי ה ASM רק עם סיומת png, למשל: ex2_q1.png, ex2_q2.png .

- **קבצי data:**
  - במקרה שנדרש להגיש בנפרד קובץ של מקטע הנתונים, שמו של הקובץ יהיה במתכונת ex2_q1_data.inc (למשל, עבור תרגיל #2 שאלה #1). יש לצרף קובץ זה לקובץ ה ZIP שאתם מגישים.
  - קובץ זה יכיל **הגדרות נתונים בהתאם למוגדר בשאלה. התכנית צריכה לעבוד נכון עם ערכים שונים של הנתונים.** כלומר, בודק התרגיל, עשוי להשתמש בקובץ דומה, אבל עם ערכים שונים, והתכנית (כמובן) צריכה לעבוד נכון.

- במקרה שדרושים לכם **נתונים נוספים, שאינם מוגדרים בשאלה**, למשל, משתנים לתוצאות ביניים, **עליכם להגדיר קטע data נוסף, בתוך קובץ ה-asm.** שלכם. כמו כן, מחרוזת ההדפסה שכוללת את שמכם ותעודת הזהות, צריכה להיות בתוך ה asm. ולא בתוך קובץ ה data שעושים לו include.

<u>קווים מנחים, לתשומת לבכם, המשפיעים על הניקוד:</u>

1. <u>יש לשים לב להוראות בנוגע להגדרת משתנים, העברת פרמטרים ,התנהגות מבוקשת של התוכניות והפונקציות ומבנה הפלט שאמור להתקבל.</u>
2. למען קריאות התרגיל יש לתעד בתחילת התרגיל מה התרגיל עושה. בתוכנית עצמה **יש להקפיד על כתיבת הערות <u>באנגלית</u>**. בעיקר בתחילת כל פונקציה ובמקום שמתבקש (כלומר לא ברור) – יש להבהיר את הכוונה.
כל <u>הגשה</u> באיחור מצריכה את אישור המרצה, **<u>ומראש</u>.**
3. **<u>איחור בהגשה ללא אישור</u>**: <u>איחור</u> בהגשה עד שבוע מפחית מהציון הסופי של התרגיל 10%. איחור מעבר לשבוע, מפחית 25%.
4. התוכנית צריכה לעבור <u>קומפילציה</u> ולהיות נכונה מבחינת syntax ולעבוד נכון!.
5. על התוכניות להישלח כ - Source Code משמעות הדבר הינה ,שיש לשלוח את קבצי ה **ASM בלבד** ולא את שאר התוצרים. כפי שצוין לעיל, יש לצרף גם צילומי מסך של ההרצה.
6. התוכניות צריכות להיות <u>קצרות</u> ולעניין – Keep it simple.
7. בתרגילים שבהם מוגדרים קטעי נתונים – אין להסתפק בבדיקת נכונות עם הנתונים שמופיעים (מוגדרים) בקטע הנתונים הנתון. **עליכם לבדוק את נכונות הפתרון שלכם עם נתונים נוספים, בהתאם לתנאי השאלה.**

<u>תרגיל זה מכיל שאלה אחת בלבד.</u>

1. רשימה מקושרת של Dos Date and Time

מבוא

בתכנית זאת אנו עוסקים ברשימה מקושרת של day-time - הרשימה מגודרת סטטית. עוברים עליה ומדפיסים/מפעילים את האיברים של הרשימה כפי שמוסבר בהמשך.

א. הגדרת day-time

הנתון העיקרי שנעסוק בו הוא ערך בגודל DWORD שמכיל שדות שמייצגים – יום בשבוע (מספר בין 1 ל- 7), שעה (בין 0 ל- 23), דקות ושניות (0-59). לפיכך השדות והביטים שמייצגים את המספר הם כלהלן:

```
bits 0 - 5   - second  (total 6 bits)
bits 6 - 11  - minute  (total 6 bits)
bits 12 - 16 - hour  (total 5 bits)
bits 17 - 19 - day  (total 3 bits)
```

ב. הגדרת מבנה הרשומה

הרשומה (struct) מכילה את השדות הבאים:

```
day_time  - DWORD
func_ptr  - DWORD
count     - BYTE  // יוסבר בהמשך
next      - DWORD
```

המטרה של שדה count תובהר בהמשך (היא קשורה לפונקציות ההדפסה-הפעלה).

ג. פונקציות ההדפסה

קיימות שתי פונקציות להדפסה, שעובדות על אותו נתון (DWORD בפורמט day_time) אך נבדלות בפורמט ההדפסה.

הפונקציה **print_1** מדפיסה את הזמן בפורמט הבא:

```
ddd hh:mm:ss AM/PM
```

כלומר, הפונקציה מדפיסה את היום בפורמט של שלוש אותיות לאחר מכן את השעה בפורמט של 01 עד 12 ולאחר מכן את הדקות ואת השניות, ולבסוף AM או PM.

לדוגמא:

```
TUE 12:05:20 AM
SAT 10:40:00 PM
```

הערה: פורמט ההדפסה של הזמן עם AM או PM צריך להיות תואם את ההגדרה בטבלה הבאה בוויקיפדיה: https://en.wikipedia.org/wiki/12-hour_clock

הפונקציה **print_2** מדפיסה הזמן בפורמט הבא:

```
full-day hh:mm:ss
```

כלומר, הפונקציה מדפיסה את היום בכתיב מלא (למשל: Saturday), לאחר מכן את השעה בפורמט של 24 שעות, כלומר השעה יכולה להיות מספר בין 00 ל 23 של שלוש אותיות לאחר מכן את השעה בפורמט של 01 עד 12 ולאחר מכן את הדקות ואת השניות, ולבסוף AM או PM.

לדוגמא:

```
TUESDAY 00:05:20
SATURDAY 22:40:00
```

<u>הערה:</u>

שתי הפונקציות האלו מדפיסות **<u>תחילה</u>** את התו CR כלומר Carriage-Return – זהו התו 13, והוא גורם לסמן לחזור לתחילה השורה. בהמשך תראו שאנחנו מגדילים את הזמן בשנייה, וחוזרים על ההדפסה. זה יגרום לכך שההדפסה הבאה תהיה על גבי ההדפסה הקודמת, ומתקבלת תחושה של אנימציה.

ד. <u>הדפסת-הפעלת הרשימה</u>

עוברים על איברי הרשימה. לכל איבר ברשימה – מבצעים את הפעולה הבאה **count** פעמים (count הוא השדה בתוך האיבר "הנוכחי"). מדפיסים את הזמן ע"י קריאה לפונקציה באמצעות המצביע func_ptr. לאחר מכן ממתינים שנייה ע"י קריאה ל SLEEP. זאת פונקציה של windows והקריאה אליה מתבצעת ע"י שתי השורות האלו:

```
push 1000 ; 1000 mili = 1 second
call sleep
```

למעשה, בקובץ ex4_q1.inc מלבד נתונים, מוגדרות מספר פונקציות. קיראו לפונקציה mySleep (פונקציה ללא ארגומנטים) והיא תבצע את ה SLEEP. שימו לב, שחייבים לשמור את האוגרים לפני הקריאה ל sleep. הפונקציה mySleep כבר דואגת לזה.

לאחר מכן, מגדילים את הזמן בשנייה אחת, ושוב מדפיסים.

כאמור – הלולאה הזאת צריכה להתבצע count פעמים.

בתום הטיפול באיבר מסוים, מדפיסים CR-LF (קריאה לפונקציה call CRLF) ועוברים לאיבר הבא.

פיתוח התכנית בשלבים

עליכם לפתח את התכנית הזאת בשלבים. זה מומלץ מאוד, ועשוי לסייע לכם מאוד בתהליך הדיבוג, ובכלל לקצר את תהליך הפיתוח.

שלב הכנה

הגדירו את הקבועים של ה STRUCT (השדות וכן STRUCT_SIZE). (יתכן שתראו שהם כבר מוגדרים בקובץ ex4_q1.inc).

שלב א

כיתבו, בידקו ובמידת הצורך דאבגו את הפונקציה print_1 ורק לאחר מכן כיתבו (בעיקר העתיקו ושנו במקצת) את הפונקציה print_2.

טיפים / המלצות:

1. שימו לב, שהדפסת מספרים מתבצעת ע"י קריאה לפונקציה writeDec. המספרים שעלינו להדפיס הם בני ספרה עשרונית אחת או שתיים. במקרה של ספרה עשרונית אחת, יש להדפיס לפני כן את התו 0. מה שאני מציע לכם – כיתבו פונקציה בשם (למשל) myWriteDec. היא תבדוק את המספר, ואם הוא קטן מ-10 היא תדפיס את התו 0. לאחר מכן, היא תקרא לפונקציה writeDec.

2. צרו מערך של מצביעים למחרוזות, כאשר המחרוזות הן SUNDAY, MONDAY,... או SUN, MON, TUE, ETC. עי"כ בכדי להדפיס את המחרוזות הדרושה, אתם ניגשים לאינדקס המתאים במערך המצביעים למחרוזות, מציבים אותו ל EDX וקוראים ל WriteString. (מוגדר בקובץ ה ex4_q1.inc).

3. פונקציות קטנות, יכולות לעשות לכם את הקוד קריא ואת הקידוד והדיבוג לקל. לדוגמה, הפונקציה הבאה:

```
print_colon PROC
  push eax
  mov al, ':'
  call writeChar
  pop eax
  ret
print_colon ENDP
```

(חלק מהפונקציות האלו מוגדרות כבר בקובץ ה ex4_q1.inc – עיינו בו בכדי לחסוך עבודה. אין חובה להשתמש בהן).

4. בידקו את הפונקציה הזאת על איבר בודד – ללא טיפול ברשימה (בשלב זה).

5. כאמור, לאחר ש print_1 עובד (מדובגת) העתיקו אותה ושנו אותה שתתאים לדרישות print_2.

שלב ב

כיתבו פונקציה שנקרא לה למשל handle_one_element – הפונקציה הזאת מבצעת לולאה של count פעמים. בכל איטרצה, היא מבצעת:

א. קריאה לפונקציית ההדפסה המתאימה
ב. sleep של שנייה אחת
ג. הגדלת הזמן בשניה.

בשלב זה עדיין לא מימשנו את פונקצית ההגדלה, אז מומלץ או לשים את הקריאה בהערה או (עדיף) לכתוב פונקציית ההגדלה "ריקה" (אולי.. היא רק מדפיסה הודעה כלשהי בכדי שנדע שאכן ביקרנו בה)

שלב ג

כתבו פונקציה שמגדילה את הזמן בשניה. מומלץ להתייחס לפונקציה incDate שראינו בשיעור (שקף 416#) ולבצע באופן דומה את ההגדלה. זכרו, שהגדלה של שניה יכולה לגרור הגדלה של דקה, והגדלה של דקה הגדלה של שעה, וזאת כשגדלה – הגדלה של היום. יום שגדל מ-7 ל-8 צריך להקטין אותו חזרה ל-1 (ראשון אחרי שבת).

שלב ד

כתבו פונקציה שמבצעת מעבר על הרשימה, ולכל איבר ברשימה קוראת לפונקציה handle_one_element.

דוגמאות קלט ופלט של התכנית

הערה מקדימה: בקובץ ex4_q1.inc מופיעה הפונקציה print_CR. פונקציה זאת מדפיסה את התו CR(להבדיל מהצירוף CR-LF). כתוצאה מכך, הזמן שמתעדכן מודפס באותה שורה כמו הזמן הקודם, דבר שנותן תחושת שעון (סוג של אנימציה). למטרות פיתוח ודיבוג, זה לא נוח, כי בעצם הפלט "דורך על עצמו". לכן, באותה פונקציה יש קריאה גם להדפסת LF, שאותה ניתן להכניס להערה או להוציא מהערה. בהתאם לכך נוצרים שני פלטים.

כאשר אנו עובדים במוד של CR בלבד (אנימציה) הפלט (הסופי) של התכנית, עבור הנתונים נראה כך (כלומר, זהו צילום המסך לאחר הרצת התכנית):

```
$==> ex4_q1.exe
Tzvi Melamed id: 012345678 Ex4-Q1
MON 05:14:57 AM
TUESDAY 18:31:01
SUN 12:00:01 AM
SATURDAY 00:00:03
SUN 12:00:03 PM
```

כאשר עובדים במוד של הדפסת LF הפלט של התכנית נראה כך:

```
$==> ex4_q1.exe
Tzvi Melamed id: 012345678 Ex4-Q1
MON 05:14:56 AM
MON 05:14:57 AM

TUESDAY 18:30:57
TUESDAY 18:30:58
TUESDAY 18:30:59
TUESDAY 18:31:00
TUESDAY 18:31:01

SAT 11:59:58 PM
SAT 11:59:59 PM
SUN 12:00:00 AM
SUN 12:00:01 AM

FRIDAY 23:59:58
FRIDAY 23:59:59
SATURDAY 00:00:00
SATURDAY 00:00:01
SATURDAY 00:00:02
SATURDAY 00:00:03

SUN 11:59:58 AM
SUN 11:59:59 AM
SUN 12:00:00 PM
SUN 12:00:01 PM
SUN 12:00:02 PM
SUN 12:00:03 PM
```

<u>הערות חשובות</u>

1) **ניתן בקלות לבזבז שעות בדיבוג של התכנית הזאת. בכדי להימנע מזה, מומלץ לעבוד בשלבים, כפי שמתואר לעיל, וגם:**

א. הגדירו היטב את הפונקציות שאתם מקודדים. לכל פונקציה:
   1. איזה ארגומנטים היא מקבלת
   2. מה סדר הארגומנטים
   3. מה גודלם של הארגומנטים
   4. הגדירו את הקבועים שבאמצעותם אתם ניגשים לארגומנטים ולשדות השונים.

ב. הרבה פונקציות "קטנות" – מאד נוח להשתמש בהעברת ארגומנט באוגר (בד"כ השתמשו ב EAX).

ג. **דבגו כל פונקציה בנפרד**. כלומר, בתהליך הפיתוח תכתבו לצורך העניין תתי-תוכניות, כ"א בודקת פונקציה מסוימת (או.. תכנית אחת, שבכל פעם אתם מוסיפים בדיקה של פונקציה בודדת). זה נראה ארוך יותר, אבל זה עשוי לחסוך לכם הרבה זמן.

ד. תוך כדי תהליך הפיתוח, השתמשו בהדפסות, שמתעדות מה קורה, מה תוצאות הביניים, וכו'. כמובן שהדפסות אלו אינן חלק מהתכנית שאתם צריכים להגיש.

<u>בדיקת שגיאות:</u>

אין צורך לבדוק שגיאות בנתונים. יש להניח שכל הנתונים נכונים, למשל, ערכים היום הם אכן בין 1 ל-7, ובהתאמה השדות האחרים.

בהצלחה!!