מיקרו-מעבדים ושפת אסמבלר 83-255

תשפ"א/2021 - תרגיל מס' 6

אחראי תרגיל - רומן

חלק 1 – שאלה 1, הפיכת הפסיקה לTSR

בחברת המשחקים הנקראת "משחקי החיים" הראו עניין רב במשחק החיים שבניתם בתרגיל בית הקודם, והביעו עניין ברכישת קוד התכנה. אך באמצע הדיונים הסוערים אודות הרכישה, התפרץ לחדר מנהל הפרסום, ודרש שבנוסף לפעילות התקינה של המשחק, הוא גם יתקין תולעת פרסומית ביציאה מהתוכנה.

התולעת הפרסומית היא משפט הנועד לפרסם את מערך השיווק של החברה:

"Your Ad Goes Here"

המשפט צריך להבהב במרכז המסך פעם ב3 שניות.

בנוסף מומלץ אם הוא גם יהפוך את צבעו, אך מנהל הפרסום לא התעקש על דרישה זו, ולכן היא לא חובה.

בזמן הרצת התכנה שלנו, רגע לפני היציאה, קוד הצגת הפרסומת מושתל למערכת כTSR, וכ3 שניות לאחר היציאה מהמשחק, הפרסומת צריכה להתחיל להופיע על המסך.

כאשר הפרסומת נעלמת אחרי ההבהוב, היא משאירה אזור שחור אחריה. במידה והיה טקסט כלשהו על המסך באזור שבו הוצגה הפרסומת, לאחר היעלמות הפרסומת הוא נשאר שחור, כאשר הטקסט בעצם מחוק. וזאת על מנת שנוכל להשאיר רושם לאורך זמן על המשתמש ולתת לו הזדמנות לחשוב על תוכן הפרסומת.

<u>שאלות עיונית</u>

1) אם נריץ את התכנית המון פעמים ברצף, האם לדעתכם תהיה בעיה כלשהי? אם כן, מהי?
2) האם לדעתכם יש דרך לפתור בעיה זו? האם ניתן לערוך איכשהו את קוד התכנית על מנת להתמודד עם בעיה זו? (אין צורך לשנות את הקוד)
3) בתור מפתחי תכנת אנטי-וירוס, כיצד הייתם מתמודדים עם התולעת? מה הייתם עושים כדי לאתר אותה? מה הייתם עושים כדי להסיר אותה? (או לפחות לגרום לפרסומת להפסיק להופיע)

רמזים והנחיות:

- שימו לב שכאשר המעבד נכנס לפסיקת הטיימר אוגר הDS נשאר להיות מכוון לסגמנט הנתונים של התוכנה **שרצה כעת** (תכנה אחרת שהיא לא התכנה שהשתילה את הTSR). אם יש צורך, חשבו מחדש על השאלות המודגשות בחלק 1 (בפרט איפה המקום הנכון לשמור את הנתונים), כאשר אתם כעת זוכרים שהתולעת רצה כ-TSR ואולי יש תוכנה אחרת ברקע.

The Program Segment Prefix (PSP) – חלק 2

Immediately before a program is loaded into the memory for execution, DOS first builds up a **P**rogram **S**egment **P**refix (PSP). The PSP contains lots of information which is mainly used by the OS but some of it useful may also be useful for the user.

The PSP is 256 bytes long and contains the following information:

```
Offset  Length  Description
0       2       An INT 20h instruction is stored here
2       2       Program ending address
4       1       Unused
reserved by DOS
5       5       Call to DOS function dispatcher
0Ah     4       Address of program termination code
0Eh     4       Address of break handler routine
12h     4       Address of critical error handler routine
16h     22      Reserved for use by DOS
2Ch     2       Segment address of environment area
2Eh     34      Reserved by DOS
50h     3       INT 21h
RETF instructions
53h     9       Reserved by DOS
5Ch     16      Default FCB #1
6Ch     20      Default FCB #2
80h     1       Length of command line string
81h     127     Command line string
```

Some of these fields are no longer relevant for use in modern MS-DOS assembly language programming. Some interesting fields include:

- <u>The first field</u> in the PSP contains an $int\ 20h$ instruction. This instruction was once used to terminate the program execution. The program would jump to this location in order to terminate. Today we used the DOS function 4Ch instead which is much easier and safer than jumping to a location in the PSP.
- <u>The second field</u> contains a value which points at the last paragraph allocated to your program. By subtracting the address of the PSP from this value, you can determine the amount of memory allocated to your program.
- <u>Fields five through seven</u> are used to store special addresses during the execution of a program. They contain the default terminate vector, break vector, and critical error handler vectors. Although these values are normally stored in the interrupt vectors for $int\ 22h$, $int\ 23h$, and $int\ 24h$, by storing a copy of the values here, you can change these vectors so that they point into your own code. When the program terminates, DOS restores those three vectors from these three fields in the PSP.

Accessing data in the PSP:

Although the PSP is loaded into the memory immediately before your program, this doesn't necessarily mean that it appears 100h bytes before your code. Your data segments may have been loaded into the memory before your code segments which ruins this method of locating the PSP.

When the program begins to run, the DS register points to the beginning of the PSP. That is, the PSP is located at DS:0000. Note, that if your program starts with

```
mov ax, @data
mov ds,ax
```

Then after those lines, DS will point to the data segment rather than the PSP.

One way to obtain the PSP address is to use a DOS call. You can load ah with 51h and execute $int\ 21h$. The segment address of the current PSP will be returned in the bx register.

שאלה 2

מנהל התמיכה של החברה, לא התלהב מהרעיון של השתלת התולעת לתוך המשחק, וציפה מבול של תלונות מלקוחות אשר יעלו על הקשר בין המשחק לתולעת. ולכן ביקש שתוסיפו אפשרות לבטל את השתלת התולעת מתוך שורת הפקודה, על מנת שיוכל להגיד ללקוחות חשובים כיצד להריץ את המשחק ללא השתלת התולעת.

עליכם להרחיב את תוכנת המשחק, כך שיהיה ניתן להריץ אותה באחת משתי הדרכים הבאות:

C:\> GAME

or

C:\> GAME NO_ADS

כאשר האופציה הראשונה (הרצה ללא ארגומנטים) גורמת להשתלת התולעת, והרצת התכנית עם הדגל NO_ADS לא משתילה את קוד התולעת.

רמזים והנחיות:

- חישבו, באיזה מקרה צריך לסיים את המשחק בקריאה לפסיקה int 21h ואיזה מקרה צריך לסיים את המשחק בקריאה לפסיקה int 27h על סמך מה שכל אחת מהן עושה.
- כאשר אתם צרכים לספק קלט לשירות 49h של פסיקה int 21h חשבו מה שמרתם בטבלת ה IVT במהלך התקנת ה-TSR.
- השתמשו במספר וקטורי פסיקה פנויים במידת הצורך.

<u>Theoretical questions:</u>

1. What will happen to the RAM memory, if we run GAME, very large number of times.
2. Can you find a solution for the problem? (No code required)