מיקרו מעבדים ושפת אסמבלר תרגיל 5 :

אחראית תרגיל- מאיה כתר

תאריך הגשה: 23.6.2022

ההגשה בזוגות, אחד מכל זוג מגיש.

נא להוסיף בתחילת כל קובץ הערה עם שמות המגישים ותעודות זהות.

הגישו תיקיית *zip* בשם Name1_ID1_Name2_ID2.*zip* הכוללת את ואת קבצי ה*asm* הנדרשים בחלק הרטוב. ואת התשובות לשאלות העיוניות צרפו בקובץ *pdf*.

חלק יבש:

ענו בקצרה (שתיים עד שלוש שורות) על השאלות הבאות:

1) הסבירו בקצרה מה תפקידו של ה*loader*? כיצד הוא משנה את הכתובות בזיכרון?
2) ב*model tiny* מה משמעות הנחיית המהדר *org 100h*?
3) מה היא שיטת *polling*? ומה השיטה המחליפה אותה?

חלק רטוב:

חלק 1- שימוש בפסיקות

א) כתבו תוכנית המדפיסה למסך בעזרת פסיקת התוכנה *21h* את המשפט :

"It's hardware that makes a machine fast. It's software that makes a fast machine slow."
— *Craig Bruce*

נגדיר שבכל שנייה (או כמה שיותר קרוב לפרק זמן זה) יודפס תו אחד מהמשפט. על מנת למדוד שנייה אחת יש לבצע **לולאה** שלוקחת בערך 1 שניה.

כמו כן, בכל 3 תווים יש להחליף צבע לרקע המשפט. שימו לב לבחור בצבע שונה מהצבע בו נכתב המשפט על מנת שהוא יהיה קריא. מקמו את המשפט באזור כלשהו במרכז המסך.

חלק 2- מקביליות

ב) נרצה להוסיף את היכולת לשנות את קצב ההדפסה למסך. בעזרת המקשים + ו - לשנות את הקצב שבו התוכנה מדפיסה על המסך.

- לחיצה על המקש + יעלה את מהירות ההדפסה פי 2.
- לחיצה על מקש – יקטין את מהירות ההדפסה פי 2.
- לחיצה על האות *p* תעצור את פעילות ההדפסה , עד שתתקבל לחיצה נוספת על המקש שתחזיר את פעילות ההדפסה.

הנחיות ורמזים:

- יש לדגום את פורט המקלדת ע"י לולאת polling בקוד התכנית, הפועלת כחלק מהלולאה שסופרת את הזמן (קזזו בהתאם את משך הזמן לפי הזמן שלוקח לבצע **polling**, כדי שעדיין כל 1 שניה לערך תתבצע הדפסה של תו יחיד).
- שמרו בקובץ בשם *print*.asm

חלק 3- הפיכת פסיקה ל*TSR*

ג) אנחנו נרצה שעל מסך התוכנית מהחלקים הקודמים יוצג שעון אשר יציג את הזמן שחלף מתחילת התוכנית (Min:Sec:msec). יש לדאוג לכך שעבור מספר שניות זוגי יהיה צבע הכתב של השעון שונה מאשר צבע הכתב עבור מספר שניות אי זוגי.

כזכור רכיב ה (Programmable Interval Timer) PIT הינו שעון הניתן לתכנות. לשעון 3 ערוצים. ערוץ 0 של שעון זה מחובר חומרתית למעבד דרך בקר הפסיקות, כך שכל 55 מילי-שניות מתקבלת במעבד פסיקת חומרה מספר 08h.

על מנת לבצע זאת יש לכתוב קטע קוד שירוץ כל פעם שפסיקה זו מתרחשת. לשם כך אנו נכתוב ISR חדשה עבור פסיקת השעון כפי שביצעתם גם בתרגיל 4. כמו כן יש לדאוג שהפסיקה תישאר כTSR אחרי שהתוכנית מסתיימת.

הנחיות ורמזים:

- ממשו את הTSR ע"י פסיקת MS-DOS, int 27h
- שמרו בקובץ בשם clockTSR.asm
- בצעו קישור בין הקבצים clockTSR.asm ו *print*.asm על מנת שהשעון יודפס על התוכנית מהחלקים הקודמים
- ניתן להניח שהתוכנית רצה פעם יחידה בלבד. כלומר, ההרצה היא תמיד לאחר ההתעוררות והפעלת ה-DOSBOX ולא נדרש לתמוך במספר הפעלות באותו session של dosbox.

חלק 4- PSP

ד) נרצה אפשרות שהתוכנית *print*.asm תרוץ גם מבלי שיוצג השעון על המסך לכן עליכם להרחיב את התוכנית *print*.asm כך שיהיה ניתן להריץ אותה באחת משתי הדרכים הבאות:

C:\> print

or

C:\> print /NOCLK

כאשר באופציה הראשונה (הרצה ללא ארגומנטים) מוצג השעון לאורך התוכנית ובהרצת התוכנית עם הדגל **/NOCLK** לא יוצג השעון על המסך.

הנחיות ורמזים:

- מידע על PSP תוכלו למצוא בנספחי התרגיל.
- 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.
- שמרו את הקובץ המעודכן בשם pspPart.asm

The Program Segment Prefix (PSP) : נספח

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.