פרק 11
פסיקות

ברק גונן

פסיקה- הגדרה

- פסיקה (**Interrupt**) היא אות המתקבל במעבד ומאפשר לשנות את סדר ביצוע הפקודות בתוכנית, שלא על ידי פקודות בקרה מותנית (פעולות השוואה וקפיצה כגון cmp ו-jmp)

פסיקה

- למה צריך לשנות את סדר ביצוע הפקודות תוך כדי ריצה?
  - תגובה לאירועים שלא ניתן לחזות מראש
  - קבלת מידע מהמקלדת
  - קבלת מידע מהעכבר
  - קבלת אות משעון המערכת

סוגי פסיקות

- פסיקות תוכנה- **Traps**- חלק מקוד התוכנית, יזומות ע"י המתכנת.
- פסיקות חריגה- **Exceptions**- כמו פסיקות תוכנה, אבל מתרחשות אוטומטית.
- פסיקות חומרה- **Interrupts**- תוצאה של רכיבי חומרה חיצוניים (מקלדת, עכבר)
- בכל המקרים, המעבד עוצר את ריצת הקוד השוטף, מטפל בפסיקה וממשיך בתוכנית מהמקום שבה עצר.

קריאה לפסיקה

- הפקודה int, בצירוף אופרנד- מספר הפסיקה:

```
int   operand
```

- לדוגמה, הפעלת פסיקה מספר 21h:

```
int   21h
```

- כל פסיקה מפעילה קוד לטיפול בפסיקה, שנקרא Interrupt Service Routine או בקיצור **ISR**.

שלבי ביצוע פסיקה

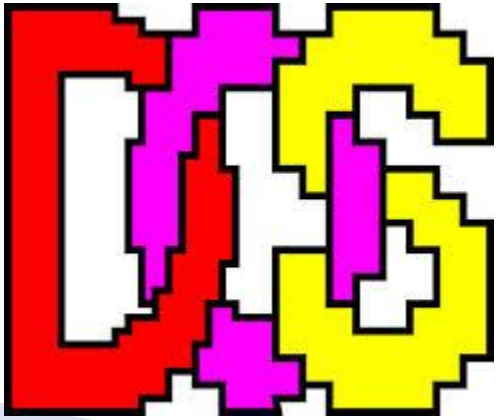
1. המעבד מסיים את ביצוע ההוראה הנוכחית
2. המעבד שומר במחסנית את תוכן רגיסטר הדגלים ואת כתובת הפקודה אליה יש לחזור בסיום הפסיקה
3. המעבד "מנקה" את דגל הפסיקות ואת דגל המלכודת (הסיבה- קריאה עצמית)
4. המעבד מחשב את כתובת ה-ISR שקשור לפסיקה ומעתיק אותה ל-cs:ip (הדרך- קריאה עצמית)
5. המעבד מבצע את ה-ISR
6. בתום ביצוע ה-ISR, המעבד מוציא מהמחסנית את הדגלים ואת כתובת החזרה
7. המעבד ממשיך בביצוע התוכנית מכתובת החזרה

Traps

- הסוג הראשון של פסיקות אותו נלמד הינו traps
- Traps הינן פסיקות יזומות ע"י המתכנת, חלק מקוד התוכנית
  - כמו פרוצדורה: התוכנית עוצרת מבצעת קוד מוגדר מראש וממשיכה מהמקום בו עצרה
- למה לא להשתמש בפרוצדורה?
  - בפרוצדורה אנחנו צריכים לקמפל את הקוד שלנו יחד עם הקוד של הפרוצדורה
  - נניח שרוצים להשתמש בפרוצדורה שהיא חלק ממערכת ההפעלה- צריך לקמפל את התוכנה שלנו עם ספריות מערכת ההפעלה- לא מעשי
- Traps הן דרך עדיפה להפעיל קוד של מערכת ההפעלה
  - כעת נעסוק בפסיקות של מערכת ההפעלה DOS

פסיקות DOS

- Disk Operating System
- מערכת ההפעלה של מיקרוסופט, קדמה ל-Windows
- מקשרת (בין היתר) בין התקני חומרה לתוכנות
- חוסכת עבודה למתכנתים
  - במקום לגשת ישירות לחומרה, מקבלים שירותים ממערכת ההפעלה
- כוללת אוסף של פסיקות, השימושית ביניהן היא 21h



int 21h

- קליטת תו מהמקלדת
- הדפסת תו למסך
- קריאת מחרוזת / הדפסת מחרוזת למסך
- קריאת השעה / שינוי השעה
- ועוד...
- איך פסיקה אחת נותנת את כל השירותים הללו?
  - הרגיסטר **ah** מחזיק את קוד השירות
  - הרשימה המלאה של קודי השירות:

http://spike.scu.edu.au/~barry/interrupts.html

קליטת תו מהמקלדת- ah=1h

- mov ah, 1h
- int 21h

- התו שייקרא יטען תוך al
- ייטען ערך ה-ASCII
  - לדוגמה אם המשתמש הקליד "2", al יכיל 32h
  - כדי להמיר מערך ASCII לספרה, יש לחסר את ערך ה-ASCII של "0"
  - sub al, ‘0’ ;sub al, 30h

הדפסת תו למסך- ah=2h

- mov dl, ‘X’ ; or any other character
- mov ah, 2h
- int 21h
- הפסיקה תשנה את ערכו של al לערך ה-ASCII של התו
- קיימים שני קודי ASCII שימושיים:
  - 0Ah- או 10- חזרה לתחילת השורה
  - 0Dh- או 13- שורה חדשה
  - הצירוף שלהם- ירידת שורה והמשך הדפסה מתחילת השורה
  - sub al, ‘0’ ;sub al, 30h

דוגמה- הדפסה למסך

- קטע הקוד הבא:
  - מדפיס X
  - יורד שורה
  - מדפיס Y

```
;print x
mov     dl, 'X'
mov     ah, 2
int     21h
;carriage return
mov     dl, 10
mov     ah, 2
int     21h
;new line
mov     dl, 13
mov     ah, 2
int     21h
;print y
mov     dl, 'Y'
mov     ah, 2
int     21h
```

תרגיל- קריאת תווים והדפסה למסך

- צרו תוכנית שמבצעת את הפעולות הבאות:
  - קוראת 2 תווים מהמשתמש
  - עוברת לתחילת השורה הבאה במסך
  - מדפיסה למסך את התו שערך ה-ASCII שלו גדול יותר

AZ

Z

הדפסת מחרוזת תווים- ah=9h

- יוצרים מחרוזת, שמסתיימת בתו '$'
  - סימן להפסיק את ההדפסה למסך
  - message db 'Hello World$'
- טוענים ל-ds את הסגמנט של המחרוזת
  - במקרה שלנו, מיותר
- טוענים ל-dx את האופסט של המחרוזת
  - mov dx, offset message
- קוראים לפסיקה
  - mov ah, 9h
  - Int 21h

קליטת מחרוזת תוים- ah=0Ah

- לימוד עצמי ☺

יציאה מהתוכנית- ah=4Ch

- קריאה ל-int 21h עם קוד 4Ch גורמת לשחרור הזיכרון וסגירת כל הקבצים שנפתחו ע"י התוכנית.
- הפסיקה מעבירה למערכת ההפעלה את מה ששמור ב-al
  - נהוג לטעון לתוך al את אפס
  - לכן יציאה רגילה מתוכנית היא
  - mov ax, 4C00h
  - int 21h

base.asm- מה למדנו?

```
IDEAL
MODEL small
STACK 100h
DATASEG
CODESEG
start:
    mov ax, @data
    mov ds, ax
exit:
    mov ax, 4c00h
    int 21h
END start
```

- אנחנו מבינים את **כל שורות הקוד** בתוכנית השלד!
  - Stack- למדנו בפרק על המחסנית
  - Int 21h- פסיקת DOS

קריאת השעה- ah=2Ch

- לצד המעבד קיים טיימר - רכיב חומרה שתפקידו לתת תזמונים
  - שולח אות חשמלי כל 0.55 מילישניות
  - "שעון 1/18 שניה"
- קריאת השעה:

```
mov     ah, 2Ch
int     21h
```

- תוצאת הקריאה:
  - השעות- ch
  - הדקות- cl
  - השניות- dh
  - מאיות השניה- dl

שימושים מעשיים בטיימר

- קריאת השעה
  - תוכנית דוגמה- timer.asm
- יצירת מספרים אקראיים
  - דוגמה בפרק על כלים לפרוייקטי סיום
- מדידת פרק זמן נתון
  - תרגיל: כיתבו תוכנית שמדפיסה למסך את הספרה 0, ולאחר שניה מדפיסה למסך את הספרה 1. שימו לב לכך שייתכן ששעון השניות ישתנה למרות שעדיין לא עברה שניה מהפעם האחרונה שקראתם אותו- לדוגמה, בזמן שהדפסתם את הספרה 0 כמות המילישניות היתה 960, לאחר עדכון של השעון אחרי 55 מילישניות בלבד ערך השניות התעדכן.
  - ;

Exceptions

- הסוג השני של פסיקות הוא פסיקות חריגה- exceptions
  - מתרחשות כתוצאה מאירוע תוכנתי חריג
- דוגמאות:
  - Int 0h- חלוקה באפס
  - Int1h- עצירה אחרי כל פקודה (step by step)
    - בלי פסיקה זו, לא היה ניתן להפעיל תוכנות דיבאגר
  - Int 3h- הרצת פקודות עד breakpoint
    - כלי חשוב בדיבאגרים

סיכום

- בחלק זה למדנו אודות סוגי הפסיקות השונים
  - Traps- ראינו דוגמאות לפסיקות DOS
    - קליטת תווים ומחרוזות
    - הדפסה למסך של תווים ומחרוזות
    - עבודה עם טיימר
  - Exceptions
  - Interrupts – פסיקות חומרה- נקדיש להן את הפרק הבא