

## מנגנון הפסיקות (Interrupts)

פסיקה היא מצב של העברת שליטה בכפיה של ה-CPU לרוטינה. דרך אחרת לחשוב על פסיקה היא הפסקה בכוח של ריצה שוטפת של תוכנית וסתעפות לרוטינה אחרת.

פסיקה היא למעשה (או יכול להיות) תגובה לאיות של החומרה שהוזע ל-CPU.

מדובר למעשה בעזירת התוכנית הנוכחית, ושבול רוטינה אחרת על פניה. הרוטינה זו תהיה בדרך כלל (אך לא תמיד) רוטינה בלתי תלויות בתוכנית שהמלכה השוטף נעצרה. הרוטינה זו "בלתי תלויות" בתוכנית המופסקת במקורו, שכוחה התוכנית (בדרך כלל) אינו מכיר את הרוטינה החלופית, אינו מעוניין בעזירת התוכנית שלו ויכול להיות שהרוטינה החלופית אינה משרתת את מטרותיו כלל. בדרך כלל התוכנית אפילו אינו מביא את האפשרות של פסיקהを考えין. המהלך של פסיקה עשוי להתרחש, בדרך כלל, אחרי כל פקודה מכונה בתוכנית, ללא תואם כלשהו אליה ולא אפשרות לחזות את נקודת הזמן או הפקודה הראשון. כמו כן שיצא מזה שבדרך כלל, ה"ROUTINES החלופיות" הללו, שבחמש נקרא להם ISR-ים, נכתבות בצורה כזו שהפעלה שלהם לא תשפיע לרעה על התוכניות המופסקות, או שהשפעה שלילית כזו תהיה מינימלית. ה-ISR בדרך כלל ישתדל להציג את השליטה לתוכנית המופסקת במהירות המרבית, והתוכנית המופסקת תתחדש במצב זהה לרגע שבו היא הופסקה.

אולי הדרך הטובה ביותר להבין את מושג הפסיקה הוא לראות איך הוא מתבטא ב-CPU:

כאשר תוכנית רגילה ממבצעת, הפקודה הבאה לביצוע הנקבעת על ידי הערך של הזוג IP:CS, מתעדכנת באופן שוטף לפי רצון התוכנית:

בעור פקודות בקרה, או הם פקודות שתפקידם לשנות את ה-IP או ה-CS:IP (כמו JMP, CALL).

בעור יתר הפקודות כמו MOV, ADD (וailleה הרוב) הפקודה העוקבת בזיכרון מיד אחרי הפקודה המתבצעת הנוכחיית.

בשני המקרים הפקודה הבאה לביצוע נמצאת בתחום התוכנית המתבצעת.

פסיקה גורמת (או עשויה לגרום) שינוי הזוג IP:CS לערכיהם שם מחוץ לתוכנית המופסקת.

## סוגי פסיקות

מבחן לוגית, פסיקות נבדלות לכמה סוגים:

1. Hardware Interrupts - פסיקות חומרה - תגובה לאירועים רכיב חומרה במחשב שהוחוץ ל-CPU. פסיקות מסוג זה הם בדרך כלל לצורכי קלט / פלט - למשל כתוכאה של לחיצת מקש במקלדת או הזנת העכבר.

2. Exceptions - חריגות - פסיקות הנובעת מ对照检查 לא תקין במהלך הרצאה של התוכנית הנוכחית. חלוקה באפס (divide overflow) היא דוגמא קלאסית לכך. דוגמאות אחרות הם ניסיון לבצע פקודות מכונה לא חוקית, ניסיון לגשת לכתובת לא קיימת בזיכרון, גישה לכונן דיסקטים שאין בו דיסקט וכו'.

3. Software Interrupts - פסיקות תוכנה - פסיקות ביוזמת התוכנית - בדרך כלל ע"י הפקודה INT. פסיקות מסוג זה הם בדרך כלל שימוש ברוטיניות שירות הקיימות בזיכרון (אך מחוץ לתוכנית). זאת כבר ראיינו: כאשר השתמשנו בספרית הרוטיניות של DOS (h 21h INT).

למעשה התאור שבראשית סיכום זה מתאים רק לפסיקות מסוג 1 ו-2 (פסיקות חומרה וחריגות). פסיקות מסוג 1 ו-2 שוניות מאוד מפסיקות מסוג 3 (פסיקות תוכנה). פסיקות תוכנה הם כמעט כמעט כמעט לרובינה: המתבנה מכיר את הרוטיניות הנקראות ופונה אליהם ביוזמתו.

### מה מבדיל בין פסיקת תוכנה להסתעפות לרוטינה (call)?

יש שני הבדלים עיקריים:

1. הסתעפות לרוטינה היא קוד הנמצא בתוך קוד ה-EXE של התוכנית הרצתה.
2. פסיקת תוכנה נעשית דרך מבנה נתונים גלובלי של המחשב.

לגביו 1. במערכות מוגנות (NT, WINDOWS, UNIX) לא ניתן באמצעות CALL לערות מה שעושים בפסיקת תוכנה: הסתעפות לקוד הנמצא מחוץ לקובץ ה-EXE. אולי יותר נכון לומר שדואגים לכך שהדבר יהיה בלתי אפשרי. כאשר תוכנית מורצת, היא

מורעתקת לזכרוֹן ומקבלת שליטה. הסתעפויות CALL מוגבלות לנקודות זכרוֹן הנמצאים ב"תמונה" הזו של קובץ EXE בזכרוֹן. כאשר תוכנית ב-C למשל קוראת לrutיניות סטנדרטיות כמו strcpy, sin, qsort (כל רוטינה שאינה קלט / פלט ואינה משפייה / מסתיעת במערכת הפעלה) מוככל בתחום קובץ EXE קוד בינהרִי (מתוך סיפריות סטנדרטיות) המשם את הפעולות הללו ונטען עמה לזכרוֹן. זה כמובן נכון גם עבור קוד מקורי שנכתב בתוכנית עצמה. רוטיניות קלט / פלט כמו fopen, fprintf הינן קודים שנמצאים בקובץ EXE אבל מעתים במערכת הפעלה ע"י פסיקות תוכנה.

תחת DOS אפשר לומר ש-CALL הוא כמעט תמיד להסתעפות לקוד בתחום קובץ EXE למקרה שאין מניעה לעשות אחרת. זה עניין של נהוג יותר מאשר עניין טכני. תחת DOS ניתן להסתעף לרוטיניות חיצונית כאילו ע"י פקודת CALL (ע"י שימוש בפורינט לפונקציה) אבל בשפות עילית שימוש כזה ב-CALL הוא נדיר מאד ונעשה רק במקרים מיוחדים.

לABI 2., קריאה לפסקת תוכנה היא דרך משאב (מבנה נתונים) גלובלי – טבלת הפסיקות IV (שיתואר להלן). המשאב הזה מוגבל (ל-256 פוינטרים) השימוש בו דורש מירומנות ואחריות ומה שאולי חשוב מכל – במערכות מוגנות (UNIX, NT) הרא גם מegan – תוכניות רגילות לא יכולות לשנות אותו.

#### מנגנון מימוש פסיקות

#### טבלת הפסיקות Interrupt Vector או IV.

במחשב PC קיימים 256 (מ-0 עד 255) מספרי פסיקות שניתן להקצות למטרות כללי ואחרות, לא כולם מנוצלות. המספר 256 הינו מאפיין של המחשב. אין אפשרות לשנות זאת. קיים מערך של 256 פוינטרים מלאים (segment + offset) שם הפוינטרים "לאן להסתעף" עבור 256 פסיקות אפשרות ממוספרות 0 .. 255. המערך טבלה זהה נקרא Interrupt Vector או IV. ב-8086 ובמצב real mode של ה-8x86 המתקדמים יותר, הפוינטרים האלו הם כולם 32bit (offset 16 .. 0 .. segment) והמערך נמצא ב-1K הכתובות הפיזיות הראשונות (1023 .. 0) של המחשב. עבור פסיקה מספר K, 4 הבטים בכתובות  $K^4$  עד  $3 \cdot K^4 + K$  הינם פוינטר מלא – קודם offset (בטים בכתובות  $1 + K^4, K^4 + 1, K^4 + 2, K^4 + 3$ ) ולאחר כך segment (בטים בכתובות  $3 \cdot K^4 + 2, 3 \cdot K^4 + 3$ ) – שהוא כתובת היעד "לאן להסתעף" בהתרחש פסיקה מספר K. היעד זה הוא לרוטינה המיוחדת שתפקידה לבצע את המזופה מהפסיקה זו, והרוטינה זו נקראת ROUTINE INTERRUPT HANDLING או ISR. לדוגמה, עבור פסיקה מספר 9 הפוינטר ל-ISR של הפסיקה נמצא בכתובות 39 – 36. ה-offset נמצא בכתובות 37 – 36, וה-segment בכתובות 38 – 39.

ב-Protected mode יש הבדלים לגבי ה-IV לעומת המצב ב-Real mode. לא ניתן לזה כאן.

### מה גורם לפסיקה?

בפסיקות חומרה - הפסיקה נגרמת ע"י איתותים של רכיבי חומרה (כמו המקלדת למשל).

בפסיקות תוכנה - ע"י ביצוע של פקודת המכונה INT.

### מנגנון מימוש הפסיקות

המנגנון מימוש הפסיקות משותף לפסיקות תוכנה וחווארה.

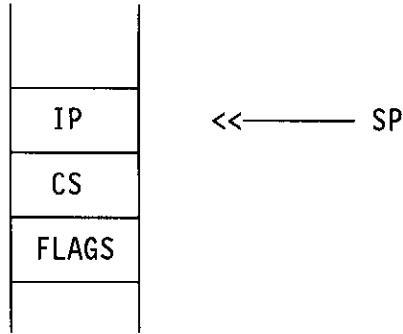
כאשר מתרחשת פסיקה מס' K מתרחש התהליך הבא:

- א. ה-CPU גומר את פקודת המכונה שהוא מבצע כרגע,
- ב. מיד לאחר מכן הוא שומר את אוגר הדגלים (FLAGS) במחסנית (פעולת PUSH).
- ג. הוא מאפס את הדגל Interrupt Flag (IF) באוגר הדגלים.

על כך נרchieb מאוחר יותר. כרגע נציין שהדבר מונע התרחשויות פסיקות נוספות, תוך שהמחשב עוסק בטיפול בנווכחות.

- ד. הוא מאפס את הדגל Trap Flag (TF) באוגר הדגלים.
- זהו דגל הקשור בעיקר למימוש debuggers. על כך נרchieb מעט בהמשך.
- ה. במחסנית נשמרים (במעין PUSH) "כתובת הפקודה הבאה" ה-CS הנוכחי ולאחר מכן ה-IP.

לפיכך תוכן ראש המחסנית היא כעת



ו. מתוך הכתובות  $K^4$  עד  $K^{4+3}$  של ה-IV נשלפים הערכים החדשים של ה-IP (K<sup>4+2</sup>, K<sup>4+3</sup>) ו-CS (K<sup>4</sup>, K<sup>4+1</sup>).

### הסתיגות

לא תמיד הכתובת CS:IP הנשמרת במחסנית היא הכתובת של הפקודה הבאה לביצוע. עבור חריגות מסוימות (כמו ה-overflow, חילוקה באפס) הכתובת נשמרת היא לא הפקודה הבאה לביצוע אלא דואק הכתובת של הפקודה שהופסקה. השיקול כאן הוא לנראה להקל על איתור תקלות.

לסיכום, מתרחש כאן תהליך דומה במידה רבה לביצוע פקודה המכונה CALL: ההבדלים העיקריים הם מקור כתובת היעד (ה-IV), שמירת אוגר הדגלים וaiפוס ה-IF.

### (IF) Interrupt Flag

בתוך אוגר הדגלים קיים דגל ה-Interrupt Flag או IF, השולט על מנגנון הפסיקות. כאשר  $IF = 0$ , מנגנון פסיקות החומרה מושבת. ה-CPU מתעלם מבקשות פסיקות חומרה. כאשר  $IF = 1$ , מנגנון פסיקות החומרה פעיל. ה-IF לא משפייע על פסיקות התוכנה (הפקודה INT). הדבר נחוץ משום שיש מצבים שבהם התוכנה לא יכולה להרשות עצמה התרחשות פסיקה, מצבים שעוד נראה.

### פקודות מכינה הקשורות למנגנון הפסיקות

CLI - איפוס ה-IF ( $IF = 0$ ).

. (IF = 1) STI - הדלקת ה-IF.

א INT - k תמיד מספר קבוע - גורם לפסיקת תוכנה k.

IRET - חזוזר מטיפול בפסיקה - שליפה (POP) מהמחסנית את ה-CS, FLAGS, CS, IP מהמחסנית שחזוזר אוגר הדגלים (לערך שישנו במחסנית) ושינוי ה-IP:CS לערcis שנשלפו מהמחסנית. שימוש לב שעם ביצוע ה-IRET, ה-IF וה-TF ישתחזרו באופן אוטומטי לערcis המקורי לפני הפסיקה עם שיחוזר הדגלים.

### ה- Trap Flag (TF)

ה-TF הוא דגל מיוחד באוגר הדגלים שכאשר הוא דלוק, ה-CPU מתפקיד בצורה מיוחדתճ מאי: הוא מבצע פסיקה (למעשה חריגה) של פסיקה מס' 1 אחרי כל ביצוע של פקודה מכונה. פסיקה מס' 1 נקראת בשל כך פסיקת ה-single step.שים לב, שמכיוון שה-TF מכובח אוטומטית ע"י מנגנון הפסיקות, התופעה זו לא תתרחש ב-ISR של פסיקה 1 עצמה. מנגנון זה נועד בעיקר למימוש תוכנות debuggerים ב-DOS. ה-debugger היה לוogh עצמו את השליטה על פסיקה מס' 1 ואחרי ביצוע של כל פקודה מכונה של התוכנית הנבדקת ה-ISR שלו של פסיקה 1 היה בודק את מצב ה-CPU והתוכנית ולפי זה היה מחליט איזה פעולות לנ��ות.

ל-TF אין פקודות מיוחדות להדliquה / כבוי שלו, כפי שיש ל-IF (CLI, STI) או IRET. למשל. ההדלקה או כיבוי הדגל נעשו בעקיפין בעזרת הפקודות POPF או PUSHF. בדרך כלל הדבר יעשה על בסיס הערך הנוכחי של אוגר הדגלים.

קוד אופיני להדלקת ה-TF הוא כלהלן:

```
PUSHF  
POP AX  
OR AX,100000000B  
PUSH AX  
POPF
```

קוד אופיני לכבוי ה-TF הוא כלהלן:

```
PUSHF  
POP AX  
AND AX,1111111011111111B  
PUSH AX  
POPF
```

### ה- ROM-BIOS

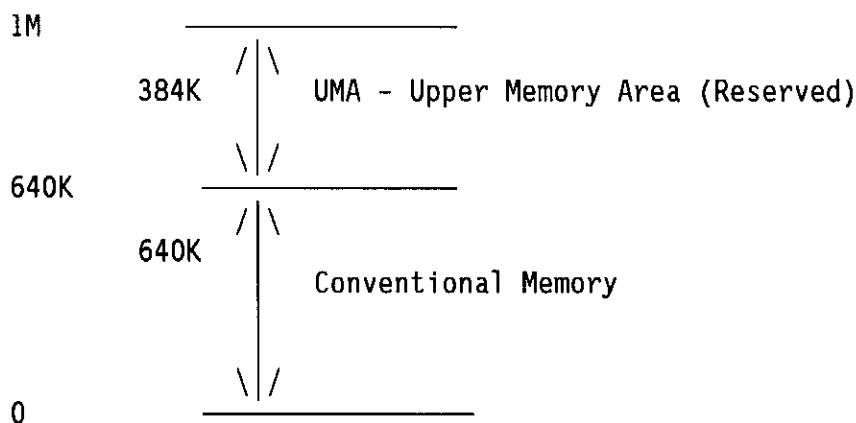
את ה- ROM-BIOS אפשר לכנסה "תוכנה באדיבות היצרן".

ראשי התיבות של ROM-BIOS הוא:  
.Read Only Memory, Basic Input Output System

מדובר בROUTINIOT ASMELI (שעקרונית כל אחד יכול לכתוב), אשר ממומשת בצוות ISR -ים. זו בעצם סיפריה רוטיניות טיפול בפסיקות.

המיוקם שלהם מבחינת כתובות זכרון הוא ב-K 384K של זכרון בין ה-K הראשוניים ל-1M או ה- UMA או ה-Upper Memory Area. אורתו שטח מיוחד שמור ל-SYSTEM כולל לא האפליקציה וגם לא מערכות הפעלה פועלם שם.

מרחב הכתובות 1M - 0 הראשונים ב-PC נראים כך:



הרוטיניות של ה-BIOS מהוים סיפריה רוטיניות שימוששות פונקציות קלט / פלט בסיסיות. רובם ISR-ים של פסיקות תוכנה אך חלקם ISR-ים של פסיקות חומרה. רוטיניות ה-BIOS מאפשרות לבוחבי מערכות הפעלה (DOS במקרה שלנו) לכתב קוד שמאצא קלט / פלט בסיסי מביי לגשת ישירות לחומרה. אחד הימורנות בכך, שלמרות שיש מספר גדול של יצרני PC ולמרות שים הבדלים בין היוצרים, DOS עורך על בולט. הכותבים של DOS לא חייב להביא בחשבון את ההבדלים בין היוצרים. במקרה-PC יהיה תואם, הוא לא חייב להיות זהה ליוצרים האחרים, מספיק רוטיניות BIOS שלו (השונות מיצורן לייצורן) יעדמו בסטנדרטים מסוימים. אלמלא היה הדבר כך, לא רק שהיו בעיות בין יצרנים, אלא גם בעיות בין מודלים שונים של אותו יצרן (386, 286, 8086, ...). במקרה לקבל תחושה מה היה קורה אללא היה BIOS, שימוש לב מה קורה כאשר מתקנים מדפסת או מודם או כרטיס רשת חדש למחשב: צריך להתקן תוכנה מיוחדת הספציפית לייצור של המכשיר החדש (הנקרא Device driver). מערכות הפעלה היום מסופקות עם דיסקים של סיפריות של

drivers של מאות יצרנים ומכשירים. זה היה המצב בעבר כל סוג של קלט / פלט, אלמלא ה-BIOS.

יש לשים לב: בנויגוד ל- INT 21h, שם חלק מערכות הפעלה (DOS) רוטינות BIOS איןם חלק משום מערכת הפעלה. הם חלק מהמחשב והם משרטים כל מערכת הפעלה שמעוניין בשירותם. מערכת הפעלה אינה חייכת, כמו כן, להשתמש בהם, חלקם או כולם, ואכן מערכות הפעלה כמו LINUX או NT בודאי מתחולמות חלק מהם, משיקולים שונים.

רשימת חלקיות של פסיקות חשובות

מספר פסיקה	שם	סוג הפסיקה	אחריות
0	Divide overflow	חריגת חומרה	DOS
5	Print-Screen	חרומרה	BIOS
8	Timer	חרומרה	BIOS
9	Keyboard	חרומרה	BIOS
(16) 10h	Video	תוכנה	BIOS
(19) 13h	Floppy	תוכנה	BIOS
(22) 16h	Keyboard	תוכנה	BIOS
(27) 1Bh	Ctrl-Break	חרומרה	DOS
(33) 21h	Function Request	תוכנה	DOS

הערות:

פסיקה מספר 21h הוא כמובן פסיקת התוכנה שהשתמשנו עד עכשו ל-קלט/פלט דרך DOS (INT 21h).

פסיקות 9 ו-16 שניהם פסיקות מקלדת (Keyboard). זו אינה כפילות. פסיקה 9 היא פסיקת החומרה של המקלדת (מתרחשת עם כל לחיצה / שחרור של מקש). לעומת זאת פסיקה 16 היא פסיקת התוכנה של המקלדת - רוטינה שתוכניות קוראות לה ביזמתם ע"י הפקודה INT 16h - על מנת לקבל קלט מהמקלדת, בדומה ל-INT 21h. למעשה INT 21h AH=1 משמש ב- INT 16h כדי לבצע את המשימה שלו.

צרייך להיות ברור, למשל, שה-ISR של פסיקת Ctrl-Break הוא פסיקה שהטיפול בה הוא באחריות DOS, שכן מדובר בחזרה למערכת הפעלה, ורק מערכת הפעלה יכולה לדעת לאן חוזרים. ה-BIOS לא יכול לדעת. וכך גם לגבי Divide Overflow.

לעומת זאת, ה-ISR-ים של פסיקות Keyboard, Timer וכו' יכולות להיות BIOS, כי מדובר בגישה להתקני חומרה, שהם חלק מחשב בלי קשר לשאלת איזה מערכת

הפעלה מותקנת.

### שימוש עקיף של ISR-ים של ה-BIOS

לפעמים מערכת הפעלה עומדת בפני דילמה: היא צריכה לחתה לעצמה את הטיפול בפסקה מסוימת (פסקה חומרה בדרך כלל) אך עדין מעוניינת להסתמך על ה-ISR של ה-BIOS בכדי לתקשר עם החומרה. דרך אפשרית להתחמוך עם הדילמה זו היא לקרוא ל-ISR BIOS מתוך ה-ISR החדש. הדרך פשוטה ביותר זאת היא ע"י שימוש ב-CALL של פוינטר לפונקציה מסווג FAR. נראה דוגמא לכך בהמשך.  
ב哀וףן כללי השימוש באמצעות זהה נראה כך:

```
PUSHF  
CALL  משתנה 32 ביט
```

כאשר המשתנה 32 ביט מכיל את הכתובת המלאה של ה-ISR של ה-BIOS.

### שימוש רוטינות טיפול בפסקה ISR

בכל הקשור לכתיבת פסקות תוכנה, אין הרבה הבדל בין פסקת תוכנה לפרוצדורה. ההבדל היחיד המתחיב מהעובדה שמדובר בפסקת תוכנה הוא, שהחזרה היא דרך הפקודה IRET (ולא RET).

בכתבת פסקה חומרה המכזב שונה מאד. פסקה חומרה מפסקה (בדרך כלל) תוכנית שאינה קשורה לפסקה. זה בודאי יכול לקרות לכל ISR חומרה. לכן ה-ISR צריך לדאוג לכך שהתוננית המופסקת לא תושפע ע"י הפסקה - התנהגותה צריכה להיות זהה למקורה שבו לא הייתה פסקה. זה כМОבן למעט מקרים נדירים. בפועל הקרייטריוון הזה מtabطا, בראש ובראשונה, בשימוש ערכים של כל האוגרים כולל אוגר הדגלים. זאת מושם שהתוכנית המופסקת משתמשת על ערבי האוגרים (מפרקות מכוונה אחת לשניה) ואין אפשרות להמנע מכך. מאחר ו-IP, CS וואוגר הדגלים נשמרים במחסנית בזמן הפסקה עצמה, ה-IRET הוא זה שמשוחרר אותו יחר, בפועל, עם אוגר ה-SP. יתר האוגרים חייבים להשמר ולהשתחרר ע"י צמדים של פקודות PUSH ו-POP בתחילת הטיפול בפסקה ובסיומה.