מבוא לפסיקות

בשיעור זה נדון בהגדרת הצורך בפסיקות, בהגדרת פסיקות ואיפיונן, ונתחיל בתאור כיצד המעבד ומערכת ההפעלה מטפלים בפסיקות.

חומר קריאה מומלץ המכיל חומר רקע חשוב:

- William Stallings, Computer Organization & Architecture, pp. 57-65
- James Feldman & Charles Retter, Computer Architecture, Chapter 3.

תיאור בתמיכה בפסיקות ב-DLX:

- שני הספרים של Muller & Paul.

1. מודלים של "מחשב"

אנו מכירים שני מודלים של מחשב: מצד אחד, מעבד DLX שמסוגל להריץ תוכניות בשפת מכונה. ומצד שני מחשב שניתן להריץ עליו ישומים שונים בו זמנית. איך בונים מחשב בעזרת מעבד DLX?

כל מערכת מחשב בנויה מהירכיה של מערכות ואמצעים לגשר בין ההירכיות. למשל, המעבד הינו רכיב חומרה אשר מסוגל להריץ תוכניות בשפת מכונה. המתכנת בשפה עילית, מאידך, סבור שהמחשב מריץ תוכניות בשפה עילית. המהדר מגשר בין רמות ההפשטה של שפת המכונה והשפה העילית.

ההפשטה המוכרת לנו כמשתמשים וכמתכנתים היא של מחשב שמסוגל להריץ בו זמנית מספר תוכניות שאפילו מורצות על-ידי משתמשים שונים. למערכת מחשב יש מספר משאבים, כגון: זכרון, דיסק קשיח, מדפסת, מקלדת, מסך ועכבר. כל תוכנית משוכנעת שכל משאבי המחשב מוקצים עבורה. יתרה מזאת המתכנת אינו צריך לדעת איזו תוכניות יורצו במקביל להרצת התוכנית שאותה הוא כותב. רמת ההפשטה הזאת מועילה הן למשתמש (שמסוגל לבצע מגוון של פעולות "בו-זמנית") ולמתכנת (שכותב את התוכנית כאילו שהיא מורצת לבדה).

ההפשטה המוכרת לנו של ה-DLX היא של מעבד שמסוגל להריץ תוכניות בשפת המכונה. יתרה מזאת, אנו יודעים כיצד ניתן לממש מעבד זה באמצעות שערים ספרתיים בסיסיים (שתיפקודם מתואר על-ידי ההפשטה הספרתית). אולם יש מספר נושאים שכלל לא עסקנו בהם, כגון: טעינה וקישור של תוכנית. דהיינו, כיצד נטענת התוכנית לזכרון? איך מותאמות הכתובות בתוכנית הנטענת לכתובות שאליהן נטענת התוכנית? ואיך יכול המשתמש להשפיע על ריצת התוכנית.

2. פערים בין המודלים

- איך מספק המעבד את ההפשטה שכל תוכנית היא הבעלים הבלעדי של משאבי המחשב (מחזורי חישוב, מקלדת, עכבר, מסך, מדפסת, מודם)?
- איך מתואמים הצרכים השונים של התוכניות המורצות בו-זמנית? דוגמאות:
  - מרחב הזיכרון: כל תוכנית סבורה שכל מרחב הזיכרון שייך לה ועומד לרשותה. תוכניות א' לא משפיעה (ללא רשות) על תוכן הזיכרון של תוכנית ב'.
  - מחזורי חישוב: כל תוכנית סבורה שהמעבד עסוק אך ורק בהרצת פקודותיה.
  - ציוד היקפי: כל תוכנית סבורה שכל הציוד ההיקפי עומד לרשותה (מסך, מודם, כרטיס תקשורת, דיסק).
- איך מאפשרים הרצת הרבה מופעים של אותה התוכנית כאשר כל מופע הוא "בן יחיד"?
- איך יכול מחשב בודד להריץ תוכניות של משתמשים שונים? המשתמשים השונים מחוברים באמצעות מסופים למחשב אחד, מתחרים על משאביו, ואינם מפריעים זה לזה (מעבר ליצירת עומס).

3. גישור בין הפערים: מערכת ההפעלה

מערכת ההפעלה היא תוכנית אשר מספקת את ההפשטה הנחוצה להרצת הרבה תוכניות בו-זמנית על-ידי משתמשים שונים. עקרון מרכזי בסיפוק הפשטה זו הוא מושג <u>התהליך</u>.

מהו תהליך?

<u>תהליך הוא מופע של ריצה של תוכנית</u>. התהליך מאפשר למערכת ההפעלה לספק לכל תוכנית שמורצת הפשטה של <u>מחשב וירטואלי</u>. הרצה של תוכנית היא הרצה על המחשב הוירטואלי ולא על המחשב הפיזי. מערכת ההפעלה מתאמת בין התהליכים (או המחשבים הוירטואליים) ומקפידה על עקרונות, כגון:

- שמירה על חוקי המשחק: כל תהליך רשאי לשחק במגרשו הפרטי, אך אסור לו לחרוג למגרשים של תהליכים אחרים. לעיקרון זה השלכות רבות: פרטיות, בטיחות ושמירת עקביות.

- הוגנות בחלוקת המשאבים: תהליך אינו יכול לתפוס משאבים למשך זמן ארוך מדי.
- מניעת מצב פט: לדוגמא: תהליך א' מחזיק במשאב A ומעוניין במשאב B. תהליך ב' מחזיק במשאב B ומעוניין במשאב A. כל אחד מן בתהליכים מסרב לשחרר את המשאבים שהוקצו לו לפני קבלת כל המשאבים הנחוצים לו להשלמת המשימה, ולכן אף תהליך אינו יכול להתקדם.

נקודה מעניינת: מערכת ההפעלה היא בעצמה תהליך. מכאן הצורך בשתי רמות לפחות של הרשאות: הרשאה של משתמש רגיל (תהליך שמריץ תוכנית משתמש, ליתר דיוק), והרשאה מיוחסת של "המנגנון" (תהליך שמריץ תוכנית של מערכת ההפעלה). תהליך של מערכת ההפעלה מקבל זכויות-יתר כדי לאפשר תפקוד כנדרש. דוגמאות: (א) מערכת ההפעלה מחליטה מתי ואיך להקצות משאבים לתהליכים. (ב) מערכת ההפעלה מפקחת על גישות לדיסק, ולפיכך מתרגמת את הכתובות הלוגיות לכתובות פיזיות.

מימוש התהליך מתבצע על ידי עטיפת כל מופע של הרצת תוכנית בסביבה המכילה נתונים שמאפשרים למערכת ההפעלה לספק את ההפשטה של המחשב הוירטואלי. העטיפה הזאת אינה נמצאת בשליטת תוכנית המשתמש, ורק מערכת ההפעלה רשאית לשנות את התוכן שלה. המתכנת אינו מודע באופן ישיר לעטיפה, אלא לדרך שבה עליו לבקש את שירותי מערכת ההפעלה (בקשת הדפסה, בקשת להציג משהו על המסך, בקשה לקרוא מה הקליד המשתמש...). במערכות עם זכרון וירטואלי, אפילו גישות לזכרון כרוכות בתרגום כתובות לפי טבלות שמנוהלות על ידי מערכת ההפעלה.

מרכיבי ההקשר של תהליך

1. מרחב זיכרון פרטי: תוכנית, נתונים, מחסנית, ערמה, מחסנית של הגרעין.
2. מצב מעבד: תוכן האוגרים הכלליים, תוכן האוגרים המיוחדים.
3. משאבים שמוקצים לתהליך: קבצים, חוצצים, וכו'.

סיבות להחלפת התהליך הפעיל

המעבד מסוגל להריץ בכל עת פקודה בודדת. (נעיר, שמעבדים, שמסוגלים להריץ מספר פקודות במקביל בכל רגע נתון, מתוכננים כך שיהיו שקולים פונקציונלית למעבד שמריץ רק פקודה אחת בכל עת, ולכן אין בפישוט זה החלשה משמעותית של הדיון.) התהליך הפעיל במהלך מחזור שעון מסוים הוא התהליך שהמעבד מריץ פקודה שלו באותו מחזור שעון (הפקודה הנוכחית נקבעת לפי הכתובת האגורה במונה התכנית). מערכת ההפעלה מאפשרת הרדמה של התהליך הפעיל והערה של תהליך אחר שנעשה פעיל.

מתי מחליטה מערכת ההפעלה שיש להרדים את התהליך הפעיל? להלן מספר דוגמאות אופיניות:

1. "שגיאה" בתהליך הפעיל: חלוקה באפס, גישה לא מיושרת לזיכרון, פקודה לא חוקית.
2. סיפוק הפשטת מרחב הזיכרון הפרטי: גישה לכתובת בזיכרון שלא ממופה לזיכרון הראשי (החטאת דף).
3. קריאה למערכת ההפעלה: בקשת הדפסה, כתיבה למסך, קריאת ההקלדה.
4. חלוקת משאבים בין תהליכים: בחלוף פרק זמן מירבי שבו מותר לתהליך להיות פעיל.
5. אירוע תלוי זמן: פעולה שיש לחזור עליה מדי פרק זמן מסוים.
6. תקלות חומרה: טעות בערך שמתקבל בגישה לזיכרון, טעות בגישה לערוץ, דיווח טעות על ידי רכיב חומרה, וכו'.
7. אתחול המחשב: לחיצה על כפתור האתחול או אתחול על-ידי מערכת ההפעלה.

מתעוררת השאלה הבאה: כיצד מופסקת הרצת התהליך הפעיל? מלבד המקרה שבו התהליך פונה לבקשה ממערכת ההפעלה (system call, trap), יש צורך במנגנון מיוחד שמאפשר למערכת ההפעלה "להשתלט" על המעבד. מנגנון זה קרוי פסיקות.

4. גישור בין החומרה למערכת ההפעלה: פסיקות

הפסיקות הן אחד מן המרכיבים המרכזיים המאפשרים למערכת ההפעלה לספק את ההפשטה של המחשב הוירטואלי לכל תהליך.

מהי פסיקה?

פסיקה היא שינוי בסדר הרצת הפקודות של תוכנית שאינו נובע מפקודות קפיצה (לרבות: קפיצה, הסתעפות, קריאה לשגרה).

רשימת חלקית של סוגי פסיקות

1. אתחול: מתרחשת על-ידי לחיצה על כפתור האתחול.
2. פקודה לא חוקית: מתרחשת כאשר התוכנית המורצת מכילה פקודה שאינה שייכת לקבוצת הפקודות של שפת המכונה.
3. גישה לא מיושרת לזיכרון: יכולה להתרחש בעת הבאת פקודה או בעת פקודת קריאה/כתיבה לזיכרון.
4. החטאת דף: הכתובת האפקטיבית שייכת לדף שאינו אגור בזיכרון הראשי.
5. קריאה למערכת ההפעלה.
6. גלישה או חלוקה באפס.
7. פסיקות קלט/פלט. פסיקות של התקנים חיצוניים למעבד כגון: מקלדת, עכבר, מודם, דיסק.

מאפייני פסיקות

1. פנימית/חיצונית:
   - פסיקה היא <u>פנימית</u> אם התרחשותה נגרמת על-ידי הרצת פקודה המופיעה בתוכנית. דוגמאות: גישה לא מיושרת לזיכרון, פקודה לא חוקית, החטאת דף, גלישה וחלוקה באפס. השם המקובל לפסיקה פנימית הוא exception, מכאן שרואים בה אירוע "יוצא-דופן".
   - פסיקה היא <u>חיצונית</u> אם התרחשותה נגרמת על-ידי מאורע חיצוני. דוגמאות: לחיצה על כפתור האתחול, הקלדה במקלדת, תקלת חומרה. השם המקובל לפסיקה חיצונית הוא interrupt.
2. אפשרות למיסוך: פסיקה <u>ניתנת למיסוך</u> אם התהליך הפעיל יכול לבקש להתעלם ממנה. דוגמאות: גלישה ופסיקות קלט/פלט.
3. חידוש התהליך:
   - פסיקה היא מסוג <u>הפסק</u> (abort) אם אין לחזור להרצת התהליך לאחר הטיפול בפסיקה. דוגמאות: אתחול, גישה לא מיושרת לזיכרון, פקודה לא חוקית.
   - פסיקה היא מסוג <u>המשך</u> (continue) אם לאחר הטיפול בפסיקה יש להמשיך בהרצת הפקודה העוקבת בתוכנית. דוגמאות: trap, גלישה.
   - פסיקה היא מסוג <u>חזור</u> (repeat) אם לאחר הטיפול בפסיקה יש לחזור ולנסות להריץ את הפקודה שבמהלכה התרחשה הפסיקה. דוגמה: החטאת דף.

5. טיפול בפסיקות

הטיפול בפסיקה כרוך בהרצת שגרה בשם "שגרת הטיפול בפסיקות". פרט למקרה אחד (פסיקה שבה מבקש התהליך שרות ממערכת ההפעלה – trap), לא מופיעה הקריאה לשגרת הטיפול בפסיקות כפקודה בתוכנית המורצת. מכאן שמנגנון הרצת שגרת הטיפול בפסיקות שונה מן המנגנון המופעל בעת קריאה לשגרה.

תזכורת: קריאה לשגרה - צילום מצב התוכנית ושיחזורו

יש דמיון רב בין קריאה לשגרת הטיפול בפסיקות ובין קריאה רגילה לשגרה. נזכר במאפיינים של שגרה רגילה:

1. פרמטרים (קלט)
2. קטע תוכנית עם משתנים מקומיים
3. ערך מוחזר בודד (פלט) .

קריאה לשגרה מתבצעת על-ידי סדרת הפעולות הבאות:

1. צילום מצב התוכנית: אחסון מצב התוכנית במסגרת ההקשר הנוכחי (של השגרה הנוכחית) במחסנית התוכנית. מצב התוכנית כולל את ערכי האוגרים הכללים, ערכי המשתנים המקומיים, מצביע לראש המחסנית, מצביעים למשתנים בערמה.
2. קריאה לשגרה: מתבצעת על-ידי אחסון כתובת החזרה באוגר R31 וקפיצה להתחלת השגרה.
3. הרצת השגרה: הרצת השגרה מתחילה בסדרה של פקודות שאינן נכתבות במפורש כאשר מתכנתים בשפה עילית. סדרת פקודות זו מיוצרת על-ידי המהדר והיא כוללת בניית מסגרת חדשה של ההקשר הנוכחי לצורך אחסון המשתנים המקומיים וכתובת החזרה. לאחר מכן מורצות הפקודות של השגרה.
4. חזרה: בסיום ריצת השגרה משוחזר מצב המעבד לפי המסגרת במחסנית שמוצבעת על-ידי המצביע לראש המחסנית. בתום השחזור, הערך המוחזר נרשם באוגר מוסכם (למשל R1) ומתבצעת קפיצה לכתובת החזרה.

חשוב לזכור! כל הפעולות הנחוצות להרצת שגרה מקודדות על-ידי פקודות בשפת מכונה.

השוואה בין טיפול בפסיקה לקריאה לשגרה

1. קריאה לפסיקה אינה ממומשת על-ידי פקודות מכונה בלבד. מצד אחד, טיפול הפסיקה כרוך בקפיצה לשגרת הטיפול בפסיקות. אולם, התרחשות הפסיקה היא בלתי צפויה (למעט פסיקה של קריאה לשגרה של מערכת ההפעלה – trap). יתרה מזאת, לפי הגדרת הפסיקה, הקפיצה לשגרת הטיפול בפסיקות אינה מופיעה כפקודה בתוכנית. מסקנה: הקפיצה לשגרת הטיפול בפסיקות מתבצעת על-ידי החומרה לאחר זיהוי התרחשות הפסיקה.
2. משיקולי יעילות, נרצה לקצר את משך הטיפול בפסיקות. לפיכך, צילום המצב ושחזורו מצומצמים למינימום ההכרחי.
3. חלק מצילום המצב ושיחזורו מתבצע על ידי פקודות מכונה שכתובות בשגרת הטיפול בפסיקות. קטעי תוכנית זו הם קריטיים ואסור שהרצתם תופרע על ידי פסיקות כלשהן. מכאן הצורך במימוש קטעים קריטיים.

דוגמא: השוואה בין קריאה לשגרה ובין קריאה למערכת ההפעלה

פסיקת קריאה למערכת ההפעלה קרויה trap או system call. פסיקה זו יוצאת מן הכלל שכן קריאתה מתבצעת על-ידי פקודה מיוחדת המופיעה בגוף התוכנית. הטבלה הבאה ממחישה את ההבדל בין קריאה רגילה לשגרה לבין קריאה למערכת ההפעלה.

סוג השגרה הנקראת	*מימוש ע"י פקודת*	*מתי מצולם המצב?*	*אפוא מאוחסן המצב?*	*בידי מי השליטה?*
שגרה רגילה	Jal, jalr	PC נשמר ע"י הפקודה. צילום שאר המצב נשמר אחרי הקפיצה, אם בכלל, ע"י פקודות DLX מפורשות בשגרה.	במחסנית התוכנית	נשארת בידי התוכנית
שגרת הטיפול בפסיקות	Trap	PC ו-MAR נשמרים לפני הרצת הפקודה הבאה. ISR אחראי לשמירת שאר המצב ולשחזורו.	במחסנית הפסיקות	עוברת לידי מערכת ההפעלה

הסיבה המרכזית להבדלים הללו נעוצה בעובדה שהמתכנת אינו אמור לדעת מהי כתובת הקפיצה לשגרת הטיפול בפסיקות או כל כתובת אחרת ששייכת לאזור המיוחס של מערכת ההפעלה. יתרה מזאת, התוכנית שמורצת בדרך כלל בהרשאה של משתמש אינה מסוגלת לקפוץ לשגרת הטיפול בפסיקות שנמצאת באזור מיוחס. לפיכך, פקודת קפיצה לאזור המיוחס גורמת לניטרול התוכנית על-ידי מערכת ההפעלה. מכאן, שהדרך היחידה לקרוא למערכת ההפעלה היא על-ידי בקשה לפסיקה! למגבלה זו יש גם יתרונות רבים. היא משפרת את המודולריות של מערכת מחשב: המתכנת אינו צריך להכיר את רזי מערכת ההפעלה כיון שאפשר להזמין שרות ממערכת ההפעלה על ידי פקודות "סטנדרטיות".

6. שתי בעיות מרכזיות בטיפול בפסיקות

1. התרחשות של יותר מפסיקה אחת בו-זמנית. באיזה פסיקה צריך לטפל ראשונה? איך זוכרים ששאר הפסיקות טרם טופלו? האם צריך לטפל בהן בהמשך?
2. התרחשות של פסיקה חדשה בעת הטיפול בפסיקה קודמת (קינון של פסיקות). האם צריך להתעלם מהתרחשות הפסיקה החדשה? האם צריך להפסיק את הטיפול בפסיקה הישנה ולטפל בפסיקה החדשה?

נפתור בעיות אלו על ידי הגדרת עדיפות לכל פסיקה. העדיפות מאפשרת לקבוע את סדר הטיפול בפסיקות הממתינות לשרות.
דוגמאות:

1. בזמן הבאת פקודה מתרחשות שתי פסיקות: גישה לא מיושרת והחטאת דף.
2. בזמן טיפול בפסיקת קלט/פלט מתרחשת פסיקת החטאת דף.

מתן העדיפויות מבטיח הימנעות ממצב פט. שגרת הטיפול בפסיקות צריכה לקיים מספר דרישות נכונות, למשל: בעת טיפול בפסיקה לא מתרחשת פסיקה של גישה לא מיושרת לזיכרון.