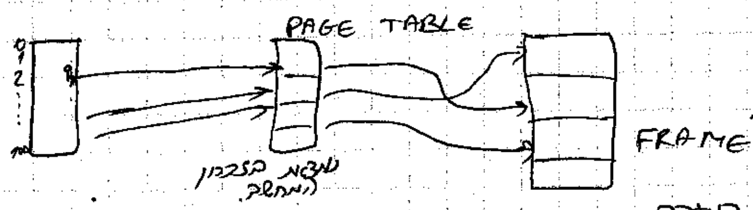
מערכות הפעלה - שיעור 5

גודל המקסימלי 1- 2^{32} לזכרון. האם נכון תמיד?

כשמשתמשים באופן יעיל ניקח 32 ביט. עקרונית ניתן להשתמש בזכרון בגודל אחר, אבל זה יעלה לנו בתקורה.

2^{32} ← 4 ג'יגה זהו חסם לגודל הזכרון בגלל גודל הכתובת. (משיקולי של גודל הכתובת.)

6 ג'יגה - אנשים שקבעו את מערכת ההפעלה בחרו זאת, כי הם חשבו שזה הכי אופטימלי. לקחו 10% מהמקסימום שזה 12 (עד 3 תהליכים) (נעשה משיקולים סטטיסטיים.)



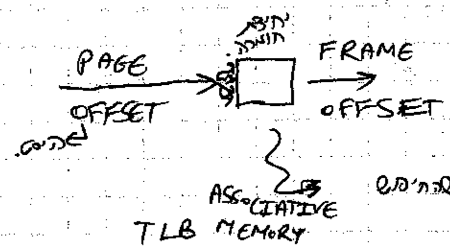
כדי לדעת את גודל ה-frame היינו מסתכלים על המעבד שמחזיק אותו. תרגום כתובות ← עקרונית לא צריך חומרה. מעשית כן צריך חומרה, כי זה מייעל.

כשרוצים לעשות הרבה פעמים ומהר משהו ← צריך מימוש בחומרה.

את הפקודה נבקש לטבלת הדפים. כל פקודה בזיכרון עיתה משהו, כי הטיפול בפקודה לפעמים חייב לעבור דרך טבלת הדפים בחומרה.

ל-cpu יש אוגר ששומר איפה נמצאת טבלת הדפים. יש גם יחידה לניהול של הזכרון ← MMU
Memory management unit

כל גישה לזכרון יש עוד גישה לזכרון כי הולך גם ל- page table מגדילים את הזמן נצטרך לטבלה ששמה יעילות יותר.



(שומרים.) מעשה זה כמו page table, אלא שהחיפוש בה צריך להיות מקבילי.

הדפים משתנים וזה לא יכול להיות ממוין, ובגלל שזה משתנה ולא ממוין יש חיפוש.

שומרים טבלה של page ומוצאים את ה-פריים.

איך נממש חיפוש מקבילי? עם חומרה. נצטרך את הסכום בכל המקומות ויהיו לנו מעגלי השוואה.

היינו רוצים לעשות זאת בתוכנה? כן, במחשבים הTLB שלנו הרבה יותר איטיים.

הבעיה הזה - החומרה המקבילית יקרה! אז יש TLB מספיק גדול לא בכל PAGE TABLE, לכן ה-TLB קטן מה-page table כי זו חומרה יקרה.

לדפים יש גודל משתנה

1. אם עובדים כל פעם ב- 8_k והדפים בגודל זה ← מצויין. לא טוב לנו דפים קטנים כי אז נעבוד עם רזולוציה קטנה יותר ← נעבוד עם גודל אחר. ככה יש לנו גודל משתנה וזה טוב.
2. תסכוף תקורה! יש הרבה חישובים. אז מה אני עובדת עכשיו.

הדרך היא טבלה לטבלת דפים ← דפה שטוחה. נשתמש במערך בגודל מס' הדפים. ובכל תא נחזיק מצביע. ביט valid או Valid חו. זה נותן לנו מנגנון הגנה של הזיכרון.

הבעיה היא שגם טבלה אחת! 4 גיגה ← טבלה של 2 מגה. בעיה לשמור זאת בנסיה בכלל היא לזיכרון גדול יותר ← טבלה גדולה יותר

האם טבלת הדפים מייצגת לנו את המצב האמיתי בכל עת? כי אז מה התועלת ב-TLB?
כי אם לא, דבר מאשר גם את טבלת הדפים זה לא יעיל.
מתי צריך לעדכן את ה-page table?
לעיתים כאלה מעבירים אותו בזכרון... זה קורה בתדירות מסויימת או על פקודה.
ההבדל בין ה-TLB ל-page table. לא נרצה של 6 גישה בזכרון לגשת ל-page table, אבל ה-TLB וה-page table משתנים.

TLB - מאיץ גישה לזכרון. זה קורה לעיתים רחוקות והתוכנה יכולה לטפל בזאת.
ה-page table משתנה רק כשמכניסים או מוציאים דף.

זיכרון פיזי - 2 מגה ← תיאורטי 4 גיגה.
סביר מאוד, תמיד כמעט שלא נשתמש בכל הזיכרון של 4 גיגה. הגבול התיאורטי של 4 גיגה הוא רק גבול. "תכנון" מעין "חורים" בזיכרון

מיפויים של הטבלאות בדפים (טבלאות דפים היררכית דו-שלבית) - היתרון הוא במקום, והחיסרון הוא בזמן. נצריך את העבודה של ה-MMU. החיפוש יותר מסובך. אלה לנו מבחינת החומרה

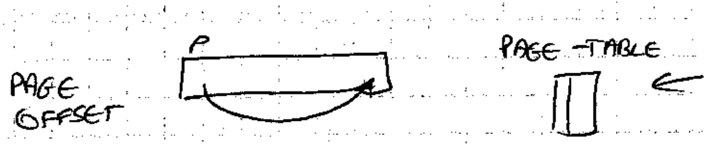
אם תהליך אופייני הוא לא בכלל לא טוב לנו ליצור היררכיה כזו

עוד דרך - טבלת דפים הפוכה: לכל דף, תהליך איפה בזכרון, נחזיק בכל מקום בזיכרון מאיפה הוא מגיע.

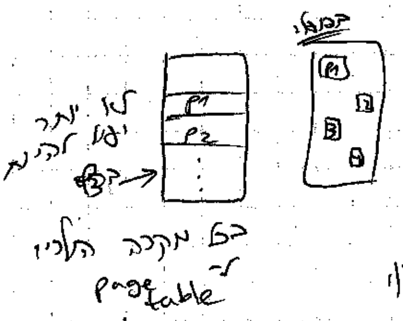
יש הרבה כניסות לאותו מקום - כי ככה זה בטבלה. חישבנו Hash וצריך לבדוק את באותו המיקום שבטבלה יש לנו את התהליך שלו באמת. נבדוק בכניסה את מס' הפריים והתהליך.

השיטה חוסכת הרבה מקום, אבל בחוץ משתמשים כי החיפוש פה די מסובך
<u>טבלאות דפים אפשריות:</u>
1. טבלה שטוחה. 2. היררכיה דולה. 3. טבלה הפוכה.

טבלת הדפים היא דינאמית, ניקח משהו ונכניס לזכרון.
מערכת ההפעלה כל הזמן יש ביקוש. היא יודעת את המשאבים לאלגוריתמים הקיימים.
אם צריך לשנות את טבלת הדפים, צריך גם לשנות את ה-TLB.
כל הפקודות שנוגעות ב-TLB הן פקודות מיוחדות, רצות רק כאשר ב-kernel mode

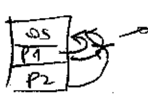


בדיקה איך אפשר לגשת.
כל תהליך יכול להשתמש רק במיפויים שלו. מערכת ההפעלה יכולה לומר "זה לא הדף שלך". אלא אם כן זה kernel mode

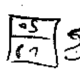


זה לא משנה אם שני דפים סמוכים או לא, לכן סדרתיות לא היתה עוזרת לי כי בכל מקרה הייתי הולכת ל-page table.

כל הזמן מבקשים "טובות" ממערכת ההפעלה - הקצאת זיכרון, קריאה ממשתמש וכו'...
להחליף תהליך יוצר תקורה אז נרצה לייעל במקום הנוכחי



אני עושה של תהליך זה שלה.



כאשר לא קופצים בין התהליכים מה החיסרון לגישה? מקומון שכמעט מערכת ההפעלה. אבל למה? כי התזמון נשאר אצלה

zs

נשים חלק מהדפים של zs בזמן שמתאימים ל-p1, p2.

תרגיל 8

ניסיון לגשת למשהו שלא ניתן לגשת אליו, למשל: לדף לא ממופה. למשל תחילת התוכנית, אלא אם כן מערכת ההפעלה שמה את זה בתוך הטבלה.
ניסיון לכתוב משהו שאסור לנו.
אם יש גישה לזכרון אחר ⇐ מעיפים את התוכנית בד"כ. לפעמים יכול להיות שרק צריך לטעון את הדף. יותר סלחני.
איך מערכת ההפעלה יודעת איזה תרגיל זה?

- מערכת ההפעלה יכולה לדעת מידע זה בהשוואה לפי הדפים שמתאימים ל-process.

להביא מידע מהדיסק, איך המידע שלו מהקומפיילר? הקומפיילר עבר קודם.

הקומפיילר מייצר קובץ אחרי לאחר הקימפול וזה על הדיסק באותו זמן. כדי לגשת לזה באופן קבוע, צריך לעשות שמירה.

להביא משהו מהדיסק, הרבה מאוד זמן. לכן מערכת ההפעלה משהה את התהליך, מביאה את המסגרת, מעדכנת את טבלת הדפים וחוזרת לתהליך.

תרגיל הדף - דף שנוי שברגע לא נמצא בזיכרון - page fault, זה תרגיל שהוא מאוד יחסית.
הטכנולוגיה במצבה קצת מטעה, כי אנחנו מתכוונים ב-page fault לדברים שהם בד"כ תקינים (רק להביא מהזכרון). + exception למה שלא תקין.