SHomework 4 Dry

Due Date: Sunday, August 8th, 2025

Teaching assistant in charge: Tsvika Lazar

Important: the Q&A for the exercise will take place at a public forum Piazza only. Critical updates about HW will be published in pinned notes in the piazza forum. These notes are mandatory, and it is your responsibility to be updated. Guidelines to use the forum:

- Read previous Q&A carefully before asking the question; Repeated questions may go without answers.
- Be polite, remember that course staff do this as a service for the students.
- You are not allowed to post any kind of solution and/or source code in the forum as a hint for other students; In case you feel that you must discuss such a matter, please come to the reception hour.
- When posting questions regarding hw4, put them under the hw4 folder.

Only the TA in charge can authorize postponements. In case you need a postponement, fill in this form - https://forms.office.com/r/rXUgsLhfnw?origin=lprLink

Dry part submission instructions:

1. Please submit the dry part to the electronic submission of the dry part on the course website.
2. The dry part submission must contain a single dry.pdf file containing the following:
    a. The first page should contain the details about the submitters - Name, ID number and email address.
    b. Your answers to the dry part questions.
3. Only typed submissions will be accepted. Scanned handwritten submissions will not be accepted.
4. Only PDF format will be accepted.
5. You do not need to submit anything in the course cell.
6. When you submit, **retain your confirmation code and a copy of the PDF**, in case of technical failure. It is **the only valid proof** of your submission.

יש לנמק כל תשובה אלא אם במפורש נאמר אחרת, תשובות ללא נימוק לא תתקבלנה.

שאלה 1 – מבנה זיכרון וירטואלי

אחרי שני כישלונות לעבור את חדו"א 1, יורם החליט לפרוש מהלימודים ולהקים חברת הזנק, שמפתחת את הדור הבא של המעבדים. יורם מתכנן מיקרו-מעבד חדש בארכיטקטורה של 24 ביט: הקומפיילר שיורם כותב מתייחס למצביעים ו-int כמילה ארכיטקטונית בת 24-ביט. יורם מנסה לקבל השראה מהארכיטקטורה של x86 ומחליט להשתמש בשתי רמות היררכיה: PGD בגודל 3KB עם כניסות שמצביעות בפוטנציה לטבלאות דפים בגודל 3KB, שמצביעות על מסגרות מידע בגודל של 3KB כ"א.
כעת הוא מתקשה להמשיך ולתכנן את המנגנון של הזיכרון הווירטואלי ומבקש את עזרתכם.

הנה השאלות שיורם מציג. הסבירו את התשובה כך שיורם יוכל להבין אותה.

1. מה גודל הזיכרון הווירטואלי שאפשר למפות בארכיטקטורה של יורם?
בארכיטקטורה של יורם יש שימוש בכתובות וירטואליות של 24 ביטים. גודל הזיכרון הוירטואלי יהיה כמות הכתובות האפשריים אליהם נוכל להגיע.יש 24 ביט ולכל ביט יש שני אפשרויות ולכן גודל מרחב הזיכרון יהיה: $2^{24}B = 16384KB = 16MB$.

2. כמה ביטים יידרשו ל-offset בתוך מסגרת מידע נתונה?
כיוון שכל מסגרת מידע בעלת גודל של 3KB, כלומר מכילה 3072 בתים (1KB = 1024 bytes). נרצה שהoffset ייצג את המידע ממספר 0 עד 3071. המספר הקטן ביותר של ביטים היכול לייצג מספר זה הוא 12 ביטים ($2^{11} = 2048, 2^{12} = 4096$), ולכן נדרש ל12 ביטים עבור הoffset בתוך מסגרת מידע נתונה.

3. כמה מצביעים / int-ים נכנסים ל-PTE?
כיוון שכל טבלאת דפים בגודל של 3KB (3072 בתים), ומצביעים וint מיוצגים ע"י מילה בת 24 ביט, כלומר 3 בתים נחלק ונקבל: $\frac{3072}{3} = 1024$, כלומר 1024 מצביעים/ int-ים (כניסות) נכנסים לPTE.

4. כמה PTEs יהיה צורך למפות?
כיוון שגודל כל טבלת דפים היא 3KB, וחישבנו בסעיף א שגודל מרחב הזיכרון הוירטואלי יהיה 16MB, נראה כי כמות הPTEs שנצטרך למפות יהיה $\frac{\{space\ size\}}{PTE\ size} = \frac{16384}{3} = 5461.33$.
יצא מספר לא שלם שחסום בין 2^{12} ל-2^{13}. כמות הPTEs תהיה 5461 (נעגל כלפי מטה).

5. כמה ביטים יידרשו כדי להצביע על PTE ספציפי?
כמות הPTEs יהיה חסום בין 2^{12} ל-2^{13} לכן נצטרך $\log_2 2^{13} = 13$ ביטים **לפחות** כדי להצביע על PTE ספציפי.

6. האם אפשר לממש את הזיכרון הווירטואלי בארכיטקטורה של יורם?
לא נוכל לממש את הזיכרון הווירטואלי בארכיטקטורה של יורם. לפי החישובים מסעיפים קודמים נצטרך 13 ביטים על מנת להצביע על PTE (לא משנה באיזה דרך נחלק את הטבלאות השונות וכמה גודל אינדקסים מקסימלי נקצה לכל טבלה, תמיד נצטרך 13 ביטים

לפחות כדי לייצג טבלאות אלו) ועוד 12 ביטים נוספים לoffset נוסף. סה"כ נצטרך 25 ביטים לפחות בזיכרון הווירטואלי כדי להגיע לכתובת הרצויה, אך נתון כי יש לנו 24 ביטים כדי לייצג כתובת שכזו בזיכרון, ולכן לא נוכל לממש.

7. בהנחה שאפשר היה לממש את הארכיטקטורה (בלי קשר לתשובה הקודמת), האם היית מציע ליורם לשנות כיוון ולמה?

הייתי מציעה ליורם לשנות כיוון מכמה סיבות:

א) ראשית, הוא מקטין את הארכיטקטורה ממה שמוכר (ארכיטקטורה של 32\64 ביטים). היו סיבות ללמה הרבה מהמערכות והמחשבים עברו ממעבדים של 32 ביט ל64 ביט לדוגמא ההגדלה של מספר הביטים פתר את בעיית החוסר מקום בזיכרון ואפשר אוטימיזציה חומרתית ותוכנתית. מעבר לכך כמעט כל הכלים המוכרים לנו היום עובדים לפי הכלים הסטנדרטיים של 32/64 ביט, ויצירת אריטקטורה שונה תדרוש התאמות מיוחדות מה שיקשה על העבודה.

ב) אורך הכתובת בארכיטקטורה של יורם הינו 24 ביט, וזהו אינו חזקה של 2. חזקה של 2 מאפשרת למחשב לבצע חישובים אריתמטיים בצורה קלה יותר, bitmasking ומימושים של page tables שונים. יצירת ארכיטקטורה בגודל זה תיצור בעיות בתחומים שצוינו.

ג) נבזבז שימוש בoffset שלנו. נתון כי כל דף בגודל 3KB, והמספר האמיתי של הoffset שנצטרך יהיה קטן מ2^{12}, ולכן לא ננצל את המיפוי המירבי שהoffset נותן לנו.

שאלה 2

1) תאר את ההבדל בין page walk אחרי TLB hit ו-page walk אחרי TLB miss.
   א) אין הבדל, תהליך ה-page walk לעולם יהיה זהה בכל מקרה.
   ב) אחרי שמקלקלים את ה-TLB עם TLB hit, חייבים לבצע page walk, אחרת אין צורך.
   ג) אחרי TLB miss חייבים לבצע page walk, אחרת אין צורך.
   ד) TLB חוסך את הצורך ב-page walk ולכן אם כבר מחליטים לפנות ל-TLB, אין צורך ב-page walk.

נימוק: TLB הוא מנגנון פיזי המכיל מיפויים אחרונים של כתובות וירטואליות לכתובות פיזיות. מנגנון זה נגיש למעבד ישירות ומכיוון שהוא חומרתי הוא מהיר יותר וישיר (אין page walk) הוא מהיר יותר מאשר חיפוש בטבלת הדפים (PGD) את המיפוי המתאים.
מכיוון שה-TLB מוגבל בגודלו, ייתכנו TLB miss – לכתובת המבוקשת אין מיפוי ב-TLB. אם המיפוי לא נמצא בלית ברירה יידרש המעבד לחפש מיפוי ב-PGD על ידי page walk.
אם מצאנו את המיפוי ב-TLB אין צורך לחפש ב-PGD שכן מערכת ההפעלה דואגת לכך שהמיפויים הרלוונטיים ב-TLB יהיו תקינים.

2) מה ההבדל בין מסגרות בגודל 4KB בארכיטקטורה 32-ביט ו-64-ביט?
   א) בארכיטקטורה של 64-ביט יש כפליים ביטים ולכן כפליים כניסות בכל מסגרת.
   ב) בארכיטקטורה של 64-ביט יש מחצית ממספר הכניסות בכל מסגרת
   ג) כל עוד מדובר באותו הגודל של המסגרת, אין הבדל בין המסגרות בשתי הארכיטקטורות
   ד) זה תלוי בדגלים שמוגדים עם עליית המחשב

נימוק: מסגרת מוגדרת כמקטע בזיכרון הפיזי (RAM). בהינתן שגדלי המסגרות זהים בין הארכיטקטורות לא יהיה הבדל בין המסגרות. המסגרת היא גודל שמוגדר למיפוי ולכן כל ארכיטקטורה יכולה להחליט מה המיפוי מדף למסגרת (4KB סטנדרטי, huge pages וכו'). ההבדל יכול להיות באופן המיפוי מכתובת וירטואלית לכתובת פיזית, לדוגמא במספר הרמות בטבלת ה-PGD (ב-32 ביט ישנן 2 רמות, בכל רמה יש טבלה של 1024 כניסות, לעומת 64 ביט, ישנן 4 רמות כאשר בכל רמה טבלה של 512 כניסות).

3) אם המסגרות והדפים הם באותו הגודל (4KB), מדוע צריך 12 ביט בשביל ה-offset ורק 10 ביט לציון הדף המתאים?
   א) צריך 12 ביט בשביל הדגלים ולכן כבר משתמשים בזה גם בשביל ה-offset. בסה"כ בזבוז של שני ביטים
   ב) בגלל הדגלים, לא נותרו מספיק ביטים בשביל הדפים ולכן משתמשים רק ב-10 ביט.
   ג) לא באמת צריך 12 ביט, אבל נותרו 12 ביט ולכן משתמשים בהם, אחרת 10 ביט היו מספיקים גם כן.
   ד) יש פי 4 יותר כניסות במסגרות המידע לעומת הכניסות בדפים בשתי ההיררכיות ולכן צריך עוד 2 ביטים.

נימוק: במערכות 32 ביט, בהינתן דף בגודל 4KB, נדרש ל-20 ביטים סך הכל על מנת למצוא את הדף (מיפוי בזיכרון aligned ל-4KB לכן ה-12 ביטים התחתונים הם offset). 10 הביטים העליונים מהווים את ה-offset ב-PDE – טבלה בגודל 4KB שבהתאם מכילה $1KB = 1024 = 2^{10}$ כניסות לכתובות הפיזיות של הטבלאות PTE ברמה השניה. באופן דומה, כל טבלה ברמה השנייה בגודל 4KB ומכילה גם היא 2^{10} כניסות. כל כניסה מתאימה למיפוי הפיזי של דף למסגרת בגודל 4KB. 12 הביטים הנותרים מתאימים ל-offset במסגרת עצמה: $4KB = 2^{12}$.

4) מה קורה בזמן שמשתמש מבקש זיכרון ממערכת ההפעלה באמצעות malloc?
   א) מערכת ההפעלה תעדיף להקצות מקום מתוך המקום שמוקצה למחסנית ואם לא נותר מספיק מקום באזור המחסנית, היא תייצר page fault
   ב) מערכת ההפעלה תגדיל את אזור המחסנית אם יהיה בכך צורך
   ג) ההקצאה נעשית מתוך זיכרון הערימה של התהליך.
   ד) מערכת ההפעלה "משאילה" מקום מתהליכים שרצים באותו הזמן ושאינם זקוקים לכל הזיכרון שהוקצה להם

נימוק: האזור שממנו מוקצה הזיכרון הינו אזור זיכרון הערימה של התהליך שהוקצה בזמן יצירת התהליך. הערימה שמורה ברשימת אזורי הזיכרון של התהליך כשדה vm_area_struct. הרשימה נמצאת ב: task_struct->mm->mmap.

אין אנו מקצים אזור זיכרון חדש. הקצאת אזור זיכרון מתבצעת על ידי mmap.

במקרה שהזיכרון הפנוי בערימה נגמר, יבקש התהליך ממערכת ההפעלה (יבוצע בתוך malloc) להגדיל את זיכרון הערימה.

5) כשתהליך רץ ויש פעולות על משתנים
   א) אם המשתנים אינם נמצאים בזיכרון ואין בהם צורך תכוף, אפשר לבצע את החישובים ישירות בדיסק
   ב) אם המשתנים אינם נמצאים בזיכרון, page fault דואג להביא אותם לזיכרון. אם משנים את ערכם, כותבים מיד את הערכים לדיסק, כדי שלא ייווצר מצב שהערכים בדיסק ובזיכרון הם שונים.
   ג) שינוי של הערכים גורם להדלקה של ביט שמציין זאת. אין צורך לעדכן מיידית את הדיסק, כי ייתכן ויהיו עוד שינויים נוספים בהמשך.
   ד) גם אם המשתנים בזיכרון וערכם אינו משתנה, בזמן החלפת תהליכים חייבים לרשום אותם חזרה לדיסק.

נימוק: שינוי של הערכים יגרום להדלקת ביט ה-dirty בטבלת הדפים (PGD) וכן בטבלת המסגרות. במידה ולא מדובר בקובץ פתוח, ההחלטה האם ומתי לכתוב לדיסק תתבצע בהתאם להחלטת אלגוריתם PFRA, האלגוריתם ידע לפנות את המסגרת לדיסק ולא למחוק אותה (ל-swap area או לכתוב אותה בחזרה לקובץ אליה שייכת).

אם מדובר בקובץ, אנו לא נכתוב לדיסק מיד (write through) אלא נשתמש במנגנון write-back שיעדכן את המידע בדיסק רק כל מספר כתיבות לזיכרון (אם מדובר בקובץ). זאת מכיוון שלפי עקרון הלוקליות בזמן אם ניגשנו למסגרת מסוימת סביר שניגש אליה בשנית בזמן הקרוב.

מעבר לכך, כתיבה לדיסק היא פעולה איטית ולכן על מנת לשמור על נצילות גבוהה ננסה להימנע מכתיבות שלא לצורך.