עבודה להגשה מס '5

~ ניהול זיכרון~

- קראו היטב את השאלות.
- ניתן להגיש עבודה בזוגות.
- הגשת העבודה תהיה דרך אתר הקורס במודל.
- בעמוד הראשון של העבודה יש לרשום את שמות ומספרי ת.ז. של המגישים.
- יש להגיש את העבודה בקובץ zip ובו יהיו הקבצים הרלוונטיי ם
- שם הקובץ שיוגש למערכת ההגשה יהיה מורכב מת"ז של המגיש/ים. לדוגמה:

עבור הגשה ביחיד - zip111111111.

עבור הגשה בזוג - 22222222_zip111111111.

- במקרה של הגשה בזוגות, רק אחד מבני הזוג יגיש את העבודה במודל.
- במקרים חריגים בלבד יש לפנות למרצה כדי לקבל אישור על הגשה באיחור.
- שאלות לגבי העבודה יש לשאול בפורום באתר הקורס (מודל) או בשעות קבלה של המתרגל/ת האחראי/ת **בלבד**. **אין לשלוח שאלות במייל המתרגלים או המרצה.**

עבודה נעימה!

בעבודה זו אתם נדרשים לממש מערכת המדמה מערכת לניהול זיכרון וירטואלי.
במערכת כזאת ההנחה שכמות הנתונים שהתוכנית צריכה בזמן ריצתה היא גדולה מאוד, ואין אפשרות להעתיק את כולם לזיכרון הראשי. אבל, המעבד יודע להשתמש בנתונים או לבצע הוראות רק אם הם נמצאים בזיכרון הראשי.
איך מתגברים על הבעיה?
מחלקים את הנתונים ל- m בלוקים בגודל קבו ע הנקראים דפים)למשל בגודל 4K(, ובכל רגע נתון דואגים שהדפים הנחוצים לביצוע ההוראה הנוכחית של התוכנית יימצאו בזיכרון הראשי .
כל הדפים נשמרים בזיכרון המשני. לעומתו, הזיכרון הראשי, יכול להכיל רק n דפים בכל רגע נתון.
שימו לב: m >> n .
כאשר התוכנית רוצה לפנות לדף מסוים, על מנת לקרוא ממנו או לכתוב אליו, הדף צריך להיות בזיכרון הראשי. לכן, לפני שניגשים לדף, בודקים אם הוא נמצא בזיכרון הראשי. אם הוא לא נמצא מביאים אותו מהזיכרון המשני.
בעבודה זו, נניח שלכל דף בזיכרון המשני יש אינדקס ייחודי – אינדקס מ- 0 עד m-1, והגישה לדף מתבצעת באמצעות המפתח שלו .
מהרגע שבו דף מסוים מועלה לזיכרון הראשי, התוכן שלו משתנה אך ורק בזיכרון הראשי. רק ברגע שדף מוצא מהזיכרון הראשי נעדכן אותו גם בזיכרון המשני – דוגמה תינתן בהמשך .

<u>מרכיבי המערכת:</u>

- זיכרון משני:
  נייצג דפים בזיכרון המשני על ידי מערך A בגודל m של מחרוזות) String(. כל תא במערך מייצג דף בזיכרון, כאשר A[i] שומר את התוכן של הדף שהמפתח שלו i.
- זיכרון ראשי:
  נייצג את הדפים בזיכרון הראשי באמצעות תור באורך n.

<u>ניהול התור ואסטרטגיות להחלפות דפים</u>

מכיוון שהזיכרון הראשי מוגבל בכמות הדפים, כאשר דף חדש מובא מהזיכרון המשני לראשי, אם אין מקום פנוי, צריך להחליפו בדף אחר בזיכרון הראשי. בעבודה זו נממש שתי אסטרטגיות להחלפות דפים:

- (First In First Out) **FIFO**: מחליפים את הדף הוותיק ביותר .
- (Least Recently Used) **LRU**: מחליפים את הדף שזמן הפנייה (קריאה או כתיבה) האחרון אליו הוא הרחוק ביותר (ישן ביותר)מבין כל הדפים הנמצאים בזיכרון.

<u>אתחול המערכת</u>

זיכרון משני: נאתחל את כל התאים של מערך בזיכרון המשני במחרוזות ריקות.
זיכרון ראשי: נעלה את n הדפים הראשונים מהזיכרון המשני לזיכרון הראשי לפי הסדר (A[0]…A[n-]1).

תמיכה בפעולות

אתם נדרשים לתמוך בפעולות הבאות:

1. read(index) – הפעולה מחזירה את התוכן של הדף שהאינדקס שלו הוא index.
2. write(index,char) – הפעולה מוסיפה לתוכן של הדף שהאינדקס שלו index את האות char (באמצעות שרשור לסוף הדף).
3. הנכם רשאים להשתמש ולממש במבנה עזר לבחירתכם (לא חובה) struct עבור ניהול זיכרון.

כפי שהוסבר קודם, בשתי הפעולות, (1) ו- (2), המערכת מחפשת קודם האם הדף נמצא בזיכרון הראשי. אם הדף קיים בזיכרון הראשי, היא עובדת עם הדף הנ"ל .
אחרת, המערכת מעתיקה את הדף מהזיכרון המשני לראשי (עם החלפת דפים לפי הצורך לפי האסטרטגיה שנבחרה).

דוגמה לפעולת כתיבה:

כתיבה לזיכרון המשני מתבצעת רק כאשר מוציאים דף שכתבו עליו מהזיכרון הראשי, לדוגמא:
לאחר ביצוע write(55, 'a'):
הדף המתאים בזיכרון הראשי יכיל את המחרוזת 'a', בעוד שבזיכרון המשני,A[55] יכיל את המחרוזת הריקה. רק כאשר דף אחר יחליף את דף זה בזיכרון הראשי, התו 'a' יתווסף לתא 55 בזיכרון המשני .כלל זה מתקיים גם אם בוצעו מספר כתיבות לדף.

אתם רשאים לממש פונקציה עזר לשיקולכם.

הרצת התוכנית

הרצת התוכנית צריכה להתבצע מתוך שורת הפקודה באופן הבא:

./MemoryManagement useLRU InputFileName OutputFileName m n

כשאר

- MemoryManagement הוא שם של קובץ הרצה
- useLRU – אסטרטגיית החלפת הדפים. פרמטר זה מהווה אינדיקטור לשימוש באסטרטגיית להחלפת דפים. אם useLRU=1 יש להשתמש באסטרטגיית "LRU", ועבור כל ערך שלם אחר יש להשתמש באסטרטגיית" FIFO".
- InputFileName – השם של קובץ הקלט אשר מכיל את הפקודות למערכת.
- OutputFileName – השם של קובץ הפלט אשר אליו נכתוב את הפלט של הפקודות הניתנות למערכת .
- m – גודל הזיכרון המשני
- n – גודל הזיכרון הראשי

לדוגמה,

./MemoryManagement 1 intput.txt output(LRU).txt 100 50

<u>קבצי קלט ופלט לדוגמה:</u>
מצורפים לעבודה קובץ קלט לדוגמה "input.txt", אשר מכיל את אוסף הפקודות למערכת ניהול הזיכרון שלכם. בנוסף מצורפים שני קבצי פלט output(FIFO).txt, ו- output(LRU).txt, המכילים את הפלט של המערכת בהתאם לשתי האסטרטגיות השונות בעבודה עבור m = 100, n = 50.

<u>הערה:</u> כל פעם שמופיעה פקודה print בתוך קובץ קלט, יש לכתוב תוכן נוכחי של זיכרון משני לקבץ פלט.

<u>הערות והכוונות</u>

- דוגמא לתוכנית בה מעבירים ארגומנטים בשורת הcommand line
https://www.youtube.com/watch?v=W9CV1IxIsmw
- מצורף לנוחיותכם קובץ קלט input.txt ודוגמאות של פלט שאמורים להיווצר כאשר משתמשים בשיטת LRU Output(Lru).txt וכאשר משתמשים בשיטת FIFO Output(FIFO).txt
- התחילו ממימוש בסיסי של מערכת ניהול זיכרון. בנו מערכים שייצגו זיכרון משני וזיכרון ראשי אשר יענו על קונספט דפדוף כפי שנלמדו בהרצאות ובמעבדות .
- בהמשך הוסיפו את הגישה לקבציי input ו- output .
- במהלך הכנת התרגיל מומלץ להוסיף הדפסות עזר בכל שלב.
- לשאלות יש לפנות בפורום שאלות עבודה מס' 5.

עבודה מהנה!