דו״ח סיכום לפרויקט גמר בקורס ארכיטקטורות מחשבים מתקדמות

שמות הסטודנטים

שם: אמיר חלבי ת.ז. 205718042

שם: חן גבאי ת.ז 308565340

# מבוא

* תיאור הבעיה ורקע כללי

פקודות S ו W הן פקודות הכתיבה והקריאה מהזיכרון כאשר המעבד מפעיל פקודה כזו נצטרך לגשת לרכיב הזיכרון , הגישה לזיכרון לוקחת המון זמן , לכן שומרים את הכתובות שנגשנו אליהם לאחרונה במטמון שהוא רכיב קטן ומהיר ואת הכתובות שנגשנו אליהם לפני יורדים בהיררכיה למטמון ברמה השני והשלישי ,

כל הפקודות נמצאות ב DDR שהוא מתאר לנו את הזיכרון הפיזי של הסימולטור וכאשר יגשים אילו לצורך קריאה או כתיבה , משארים את הדף פתוח כך שככל שפותחים דפים במקביל הסכוי לגשת לדף פתוח הוא יותר גדול וזה יחסוך לנו בזמן הגישה לזיכרון

* מטרות

בנית כלי סימולציה Memory hierarchy performance simulator

בחינת ביצועים של הרכיב הזיכרון לפי המבנה שבחר המשתמש

* הדרך והמצעים להשגת המטרה

יצירת קובץ קונפיגורציה שיאותחל על ידי המשתמש

יצירת trace עם מספר תוכנות בדיקה

יצירת קובץ עזר קובץ בינים שמכיל את הכתובות שניגשות לזיכרון ואת את ה סוג שלהם (קריאה או כתיבה)

יצירת מבנה נתונים כדי לשמור נתונים רלוונטיים שצרכים לכנס למטמון כולל הפונקציונליות של המערכת

יצירת מבנה נתונים שישמש לדפים פתוחים ב DDR

# תיאור הפתרון

כאשר מתקבלת כתובת לדוגמא : "W 0x0001fd0e”

* נמיר את הכתובת לבתים ( 32 בתים)
* נחלק את הכתובת לשלוש שדות Index Offset Tag ולכל מטמון ההיררכיה לאו דוקה דומים
* לכל מטמון ניצור מבנה נתונים שישמור את שדה ה Tag
* לכל מטמון ניצור מבנה נתונים שישמור את שדה ה Valid Bit
* לכל מטמון ניצור מבנה נתונים שישמור את שדה ה ) Dirty Bit במידה והמשתמש בחר ב (write back
* נחלק את הכתובת של ה DDR ל Offset ו Tag
* ניצר מבנה נתונים שמכיל את מספר הדפים המקסימלי שיפתחו במקביל
* כשאר ניגשים ל DDR שומרים את הדף שמכיל את הכתובת ואת השורה שלו
* Replacement Policy

לפי בחירת המשתמש במנגנון המיפוי של בלוק לרמה מתחת איזה בלוק להעיף ?

* + כאשר המשתמש בוחר ב RANDOM מופעלת פונקציה שבוחרת בלוק רנדומלי
  + כאשר המשתמש בוחר ב LRU התוכנית יוצרת לכל שורה בכל אחד מהמטמונים מערך כדי לשמור את רמת העדיפות לפי האלגוריים של LRU וכך הבלוק בעל עדיפות נמוכה יותר יתמפה למטמון בעל רמה נמוכה יותר בהיררכיה
  + כאשר המשתמש בוחר ב PLRU התוכנית יוצרת לכל שורה בכל אחד מהמטמונים מעץ כדי לשמור את רמת העדיפות לפי האלגוריים של PLRU וכך הבלוק בעל עדיפות נמוכה יותר יתמפה למטמון בעל רמה נמוכה יותר בהיררכיה
* Write Allocate Or write OnAllocate
* לפי בחירת המשתמש Write Allocate הסימולציה מעדכן את כל המטמונים שחטפו miss מוסיף אליהם את הבלוק שנגשנו אילו , וכאשר השורה שתמפה אליה הבלוק מחדש מליאה לכן צריך להחליט על איזה בלוק מותרים , כאן נכנס התפקיד של Replacement Policy

ב write No Allocate הסמולטור יעדכן רק את ה DDR

* Write Policy

כאשר עובדים עם Write back צרכיך להחזיק Dirty Bit לכל בלוק בכל שורה בכל מטמון וכאשר נגשים לבלוק הזה לצורך כתיבה מדליקים את ה Dirty Bit כדי שנגיד לרמות מתחת שיש מידע ברמה מלמעלה שלא מעודכנת בכל רמות הזיכרון וכאשר מוחקים בלוק צרכים לבדוק האם ה Dirty Bit דלוק אם כן לפני שמוחקים אותו מעדכנים את המידע שיש לבלוק ברמה מתחת מדליקים אצלה את ה Dirty Bit ומוחקים את הבלוק .

* Write Through
* ב Write Through כל פעם שמטמון חוטף HIT מעדכן את כל הרמות שמתחתיו ולכן לא צרכים להחזיק Dirty Bit
* NWaySetAssociativeCacheL1

המשתמש רשאי לבחור כמה בלוקים יהיו לו בכל שורה בכל אחד מהמטמונים בתנאי שהמספר שיבחר המשתמש יהיה שתים בחזקת משהוא

* NWaySetAssociativeCacheL2

המשתמש רשאי לבחור כמה בלוקים יהיו לו בכל שורה בכל אחד מהמטמונים בתנאי שהמספר שיבחר המשתמש יהיה שתים בחזקת משהוא

* NWaySetAssociativeCacheL3

המשתמש רשאי לבחור כמה בלוקים יהיו לו בכל שורה בכל אחד מהמטמונים בתנאי שהמספר שיבחר המשתמש יהיה שתים בחזקת משהוא

* L1 L2 L3 Cache Size

המשתמש רשאי לבחור את גודל של כל אחד מהמטמונים

* blockSize

המשתמש רשאי לבחור את גודל הבלוק

* Address Mapping

המשתמש רשאי לבחור את הגישה לזיכרון ה DDR

Row Interleaving

Cache Block Interleaving

* גם כן המשתמש רשאי לבחור את
  + מחזורי השעון לגישה לכל אחד מהמטמונים את רוחב ה bus
  + זמן הגישה לדף פתוח שורה פתוחה בדף
  + זמן הגישה לדף פתוח
  + זמן גישה לדף שאינו פתוח

הקוד מתחלק לשני חלקים אם הכתובת מסוג Write or Read

READ

**L1Cache read miss and L2Cache read miss and L3Cache read miss**

נגשים ל DDR מעדכנים דף ושורה פתוחים ומעדכנים את מחזורי השעון

מפעלים את Write Allocate על L1 L2 L3 ומעדכנים את מחזורי השעון ואם מעבירים בלוקים בין הרמות נתחשב ברוחב ה bus .

**L1Cache read miss and L2Cache read miss and L3Cache read hit**

נגשים ל L3

מפעילים את Write Allocate על L1 L2

**L1Cache read miss and L2Cache read hit**

נגשים ל L2

מפעילים את Write Allocate על L1

**L1Cache read hit**

נגשים ל L1וקראים ממנו את הערך שצריכים

WRITE

**L1Cache write miss and L2Cache write miss and L3Cache write miss**

* אם Write Allocate

נגשים ל DDR מעדכנים דף ושורה פתוחים ומעדכנים את מחזורי השעון

מפעלים את Write Allocate על L1 L2 L3 ומעדכנים את מחזורי השעון ואם מעבירים בלוקים בין הרמות נתחשב ברוחב ה bus .

* אם Write No Allocate

נגשים ל DDR מעדכנים דף ושורה פתוחים ומעדכנים את מחזורי השעון

**L1Cache write miss and L2Cache write miss and L3Cache write hit**

* אם Write Back

נגשים ל L3 מעדכנים את

הבלוק מדליקים את ה Dirty Bit

* אם Write Through

מעדכנים את הבלוק היררכית הזיכרון

* אם Write Allocate

מפעלים את Write Allocate על L1 L2

**L1Cache write miss and L2Cache write hit**

* אם Write Back

נגשים ל L2 מעדכנים את

הבלוק מדליקים את ה Dirty Bit

* אם Write Through

מעדכנים את הבלוק היררכית הזיכרון

* אם Write Allocate

מפעלים את Write Allocate על L1

**L1Cache write hit**

* אם Write Back

נגשים ל L1 מעדכנים את

הבלוק מדליקים את ה Dirty Bit

* אם Write Through

מעדכנים את הבלוק בכל היררכית הזיכרון

# אמצעי המחקר

אנחנו עובדים על trace-ים של תוכניות.

ה-trace generator של risc-V לוקח תוכניות מקומפלות ומייצר קובץ trace של פקודות אסמבלי של המעבד risc-V, ומשם דרך בסימולטור שאנחנו הולכים לבנות, צריך לחלץ את הפקודות הרלוונטיות ולנתח את התוצאות.

בסיום הריצה יוצגו שיעור ההחטאות ושיעור הפגיעות בכל רמות הCache – L1,L2,L3 ו ה DDR

לנתח כל פקודה רלוונטית האם היא קריאה או כתיבה ובנוסף איפה היו ההחטאות ואיפה היו הפגיעות בכל אחת היררכית הזיכרון.

וכמה מחזורי שעון לקח לזיכרון לפענח את כל הפקודות .

התשתית הניסויית הינה כלי סימולציה בשפת C++

# תיאור וניתוח התוצאות הניסוייות

תיאור התוצאות הניסוייות בצורה מפורטת של הרצת הסביבה על תכניות בדיקה כנדרש בפרויקט ובדיקת התוצאות.

יש לתאר את אופן הבדיקה של התוצאות.

ניתוח התוצאות והסקת מסקנות

// 0 is write No Allocate ;

// 1 is write Allocate

// 0 is write through

// 1 is write Back

// in 0 the Replacement Policy is Random

// in 1 the Replacement Policy is LRU

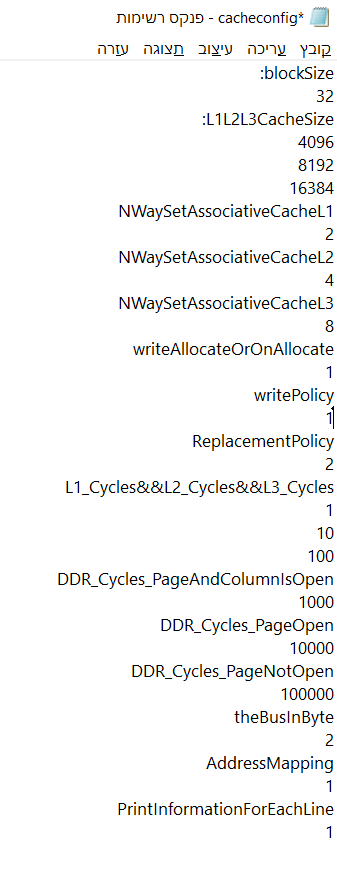
// in 2 the Replacement Policy is Pseudo LRU

// in 0 the Address Mapping Policy is Row Inter leaving

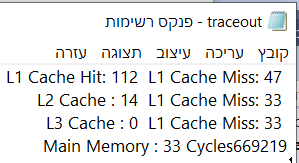
// in 1 the Address Mapping Policy is Cache Block Inter leaving

עבור קובץ קיגרצה עם הנתונים האלה :

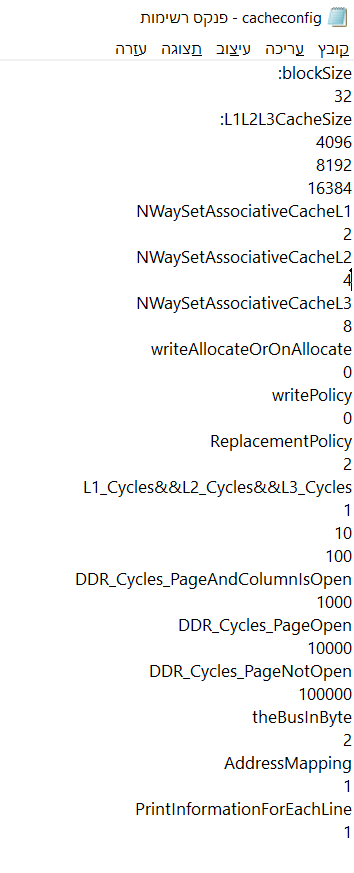
כאשר בוחרים ב write Allocate ו write Back ו מנגנון המיפוי PLRU ו Cache Block Inter leaving



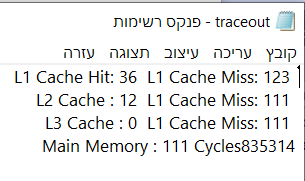
# אלה התוצאות



כאשר בוחרים ב write No Allocate ו write through ו מנגנון המיפוי PLRU ו Cache Block Inter leaving



תוצאות :



# 5.סיכום

* סיכום הפרויקט.

לפני כתיבת הקוד למדנו את הנושאים בצורה מעמיקה חזרנו על החומר של המטמון מקורס במנה מחשבים וגם ראינו שוב את ההרצאות שדברו על היררכית הזיכורן מהקורס ארכיטקטורת מחשבים

לאחר סיכום מעמיק על החומר התחלנו לתכנן את הארכיטקטורה של הסימולטור בנוסף אספנו חומרים מהאינטרנט

תכננו את המשימות בנינו כדי לעבוד במקביל על נושאים שונים

למדנו לעבוד עם קבצי טרסים ולנתח אותם

בחרנו בשפת C++ במימוש הסימולטור מכיוון שניתן לתכנת דרכה גם בתיכנות מונחה עצמים וגם בכתיבת קוד רגיל

כתיבת קוד

התחלנו לחשוב איך לסנן את הפקודות מהקוץ הקלט ולהישאר רק עם הפקודות הרלוונטית לכן יצרנו קובץ בינים שמכיל רק פקודה והסוג שלה

יצרנו קובץ קונפיגורציה למשתמש שיוכל להגדיר

גודל הבלוק

גודל המטמון L1 L2 L3

אסיוסיאטיביות L1 L2 L3

Write Allocate OR Write No Allocate

Write Policy

Replacement Policy

להגדיר את מחזור שעון של L1 L2 L3

רוחב ה bus

פרמטרי הDDR

להגדיר את מחזורי השעון של ה DDR האם דף פתוח או לא

עבדנו על כל אחד מהפרמטרים האלה חשבנו איך ואיפה לשתול את כל אחד מהתהליכים האלה כדי שבסופו של דבר הסימולטור יעבוד בצורה מסונכרנת וייתן לנו את התוצאות המתאימות

* סיכום של התרומה של העבודה שביצעתם

העמקה בנושא היררכית הזיכרון מטמון

* חשיבה ביקורתית:
  + מה היה טוב במימוש
* התוכנית דינמית יכולה להשתנות מהרצה להרצה לפי בחירת המשתמש
* שימוש בשפת C++ אפשר לנו לעבוד גם עם OOP וגם עם מבנים ופונקציות גלובליות
* שימוש במבני ניתונים שונים כגון array , victor , tree , deque , map

* + מה ניתן לשפר
  + אפשר לצור מטמונים מיקומיים שיהיו בתהליכים מקבלים ולסנכרן בניהם