|  |  |  |
| --- | --- | --- |
| **BAR-ILAN UNIVERSITY (RA)**Faculty of EngineeringRamat-Gan 52900, Israel |  **Tel: 03-5317722****engbi@mail.biu.ac.il** | **אוניברסיטת בר-אילן (ע"ר)**הפקולטה להנדסהרמת-גן 52900 |

**מבנה מחשבים ספרתיים**

**תש"ף סמסטר ב' מועד א'**

**83-301**

**מרצה:** פרופ' שמואל וימר

**מתרגל:** מר בנימין פרנקל

* **יש לקרוא היטב את ההוראות.**
* **חובה לענות על כל השאלות.**
* **ציון מקסימלי בבחינה: 100 נקודות.**
* **יש לנמק את כל תשובותיכם**.
* יש להקפיד על כתב יד קריא!
* יש לרשום תשובות בתוך הטבלאות המצורפות במקום שנדרש.
* **חומר עזר מותר בשימוש**: מחשבון, וכל חומר עזר מודפס.
* **משך הבחינה:** שעתיים**.**
* **יש לצרף את שאלוני הבחינה למחברת!**

**בהצלחה!**

**שאלה מספר 1 – Memory (65 נקודות)**

נתון מעבד בעל מרחב כתובות ווירטואלי של סיביות, רוחב מילה וזיכרון פיזי , וכן נתון כי רזולוציית המיעון היאper-byte . במעבד קיים  בגודל . גדלו של בלוק הוא וזיכרון המטמון הינו physically tagged. החלפת הבלוקים נעשית בשיטת LRU.
נתון שגודל frame הינו .

1. מהו מספר ה- sets, מהו גודל ה- tag ומהו מספר הכולל של סיביות ה- tag בזיכרון המטמון? (6 נקודות)

ה- block offset הינו 3 סיביות, ומספר ה- blocks ב- cache הינו . בנוסף, מאחר ומדובר ב- כל set מכיל בלוקים, ולכן מספר ה- sets הינו . מכאן נסיק כי גודל ה- tag הוא:
. סיביות. בסה"כ בכל ה- cache ישנן: *סיביות של* tag.

1. במהלך התכן של המעבד התברר שמסלול ההשהיה הקריטי עובר במערכת הזיכרון ואורכו , כאשר מחציתו נובע מתרגום כתובת ווירטואלית לכתובת פיזית, והחצי השני נובע מהגישה ל cache. על מנת להאיץ את ביצועי המחשב הציע מהנדס צעיר בוגר בר-אילן לצנר את מערכת הזיכרון. האם הדבר ניתן לביצוע? אם כן, כיצד? בהזנחת זמני ההשהיה ברגיסטרים ובלוגיקה, מהו התדר המרבי אליו ניתן להאיץ את פעולת המחשב? (6 נקודות)

ניתן לביצוע ע"י הפרדת התרגום מכתובת ווירטואלית לפיזית (TLB) והגישה ל cache לשני שלבים נפרדים בצינור. זמני ההשהיה בתרגום (גישה ל (TLB ובגישה לcache הינם: ועל כן התדר המרבי הינו . כמובן, כל זאת בהנחה שאין שום מסלול קריטי אחר במערכת שאורכו אמנם קטן מ- אך גדול מ- .

1. מהנדס צעיר בוגר הטכניון טען כי ניתן להגיע להאצה המושגת בסעיף ב' ללא צורך בהעמקת הצינור, וזאת ע"י גישה ל- TLB ולזיכרון המטמון באופן מקבילי. לשם כך, כדי שהגישה באופן מקבילי תוכל להתבצע, המהנדס טען כי יש צורך לבצע שינוי בגודלו של ה- frame ו/או בגודלו של ה- cache, והמליץ לבצע את השינוי דווקא בגודל ה- frame, משום ששינוי בגודלו של ה- cache יגדיל את זמן הגישה אליו. האם הוא צודק? אם כן, האם הדבר משפיע על גודלו של ה- TLB? האם ישנה דרך להימנע משינוי גודלו של ה- frame? נמקו היטב את תשובתכם. (6 נקודות)

הדבר אכן ניתן להשגה וזאת בתנאי שיהיו לנו מספיק סיביות מה- page-offset כדי לגשת את זיכרון המטמון. במקרה זה ה- page offset גדול שווה ל-cache index (ביחד עם ה- block-offset), ואין צורך בתרגום של TLB בכדי לגשת ב- cache לאינדקס המתאים. להצעתו של המהנדס אין כל השפעה על מספר הכניסות של ה- TLB, שמספרם נקבע משיקולי זמני גישה ושיקולי hit/miss rate של מערכת הזיכרון הווירטואלי. לכן, אם גודלה של כל כניסה נותר קבוע, הרי שהגדלת ה- frame איננה תשפיע על גודל ה- TLB. אמנם, שדה ה- VPN ושדה ה- PFN יקטנו עם הגדלת ה- frame, ולכן, סטודנטים שענו כי ה- TLB יקטן מהסיבה הזאת – תשובתם התקבלה, משום שלא נתון בשאלה האם גודלה של כל כניסה הוא קבוע. ישנה דרך להימנע משינוי גודלו של ה- frame, וזאת ע"י הגדלת האסוציאטיביות (דרושה דרגת אסוציאטיביות של לפחות 4-way).

שימו-לב לכשל בטענתו של הסטודנט שטען שלא כדאי לבצע שינוי בגודלו של ה- cache כי הדבר יגדיל את זמן הגישה אליו. על מנת שהתנאי יתקיים (כנראה ש: w<2) יש צורך או להגדיל את ה- frame או להקטין את ה- cache (או להגדיל את האסוציאטיביות, כאמור). הקטנת ה- cache איננה **מגדילה** את זמן הגישה אליו, אדרבה, היא יכולה רק **להקטין** את זמן הגישה. אמנם, היא תפגע כנראה ב- hit-rate של ה- cache, אך לא זו הייתה טענתו של הסטודנט.

1. יהי . בהנחה שההצעה בסעיף ג' אכן אפשרית, האם הדבר משפיע על מספר סיביות ה- tag הכולל? נמקו היטב את תשובתכם, ואם התשובה חיובית רשמו ביטוי מפורש למספרם במערכת הזיכרון הנתונה בסעיף א'. (5 נקודות)

אם נקבל את הצעתו של המהנדס מסעיף ג' ונגדיל את גודל ה- frame, לא תהיה לכך שום השפעה על מספר סיביות ה- tag בזיכרון המטמון. אמנם, אם נגדיל את מידת האסוציאטיביות של זיכרון המטמון תהיה לכך השפעה על מספר סיביות ה- tag שהוא: *סיביות של* tag *בזיכרון המטמון כולו.*

1. מהנדס שלישי בוגר אונ' ת"א הציע לוותר לגמרי על תרגום כתובות ווירטואליות לפיזיות לשם חיפוש בזיכרון המטמון. האם הדבר אפשרי? אם כן, מהן הסכנות בוויתור על התרגום? האם ניתן ע"י וויתור על התרגום להגדיל את תדר השעון מעבר להצעה בסעיף ב'? נמקו היטב את תשובתכם. (6 נקודות)

הדבר אכן ניתן להשגה וזאת ע"י הפיכת ה- cache ל- virtually indexed virtually tagged, שבו משתמשים גם בגישה ל- cache וגם ב- tag בכתובת ווירטואלית.

הסכנות שבשימוש ב-virtual tag הינן בעיות של aliasing ושל homonyms.

בסעיף ב' נאמר שזמני ההשהיה של ה- TLB ושל ה-cache שווים, ועל-כן הצעה זאת איננה משפרת את תדר השעון המרבי הניתן להשגה.

1. האם ההצעה בסעיף ה' משפיעה על גודל ה tag? אם כן, כמה סיביות ידרשו בסה"כ למימוש החדש? בהנחת אסוציאטיביות ובהתעלם מהסיביות הנדרשות לניהול ה cache (dirty וכו') מהו היחס בגדלו הכולל של ה- cache בסעיף זה לזה שבסעיף א'? האם הגדלת האסוציאטיביות מגדילה, איננה משנה, או מקטינה את היחס הזה? (6 נקודות)

מאחר והכתובת הווירטואלית הינה בת 36 סיביות ואלו הפיזית בת 32 סיביות, ה- tag של כל בלוק יגדל מ- ל- . מאחר ומספר הבלוקים איננו משתנה, וכל block entry מכיל הן data והן tag, היחס בגודל הוא:

הגדלות האסוציאטיביות תקטין את היחס הנ"ל, בהתאם לביטוי.

1. עבור נתוני סעיף א' מהו גודלו של page table? כמה כניסות ישנן בו, ומה גודלו הכולל ב- Bytes? הנח שהכתובת הפיזית יחד עם הסיביות הנדרשות לניהול הדפים מאוחסנות ב- 64 סיביות. מהו המספר המקסימלי של תהליכים היכולים להתקיים בו זמנית בזיכרון הפיזי? (6 נקודות)

מאחר ועובדים במרחב כתובות ווירטואלי, , כלומר, למערכת הזיכרון הווירטואלי ישנם דפים, ולכן זהו מספר הכניסות הנדרש ב- page table. כל כניסה מייצגת כתובת של frame שיחד עם סיביות ניהול הדף תופסות 64 סיביות, או 8 בתים. על-כן, הגודל הכולל של טבלת דפים הוא: .

ה- page tables מאוכסנים בזיכרון הפיזי שגודלו . על כן לכל היותר יכולים להתנהל בו זמנית בזיכרון תהליכים. כמובן שזהו חסם תיאורטי, משום שזה לא מותיר בזיכרון שום מקום ל- data ול- instructions.

1. על מנת שמערכת ההפעלה תנצל טוב יותר את הזיכרון ותוכל למקם את ה- page tables באופן יעיל יותר, הוחלט לפצל את הכתובת הלוגית לשלשה שדות כדלהלן: , ו , כאשר משמש את ה- outer table ו- את ה- inner tables. האם השינוי הזה מגדיל, איננו משנה, או מקטין את סה"כ גודל הזיכרון הדרוש לניהול מערכת הזיכרון שבסעיף ז'? אם הוא משתנה, בכמה? (6 נקודות)

בעוד שסכום גדלי ה- inner tables שווה בגדלו ל- page table שבסעיף ז', מתווסף כעת גודלו של ה- outer table, שהוא בסה"כ: .

1. ב- cache שבסעיף א' נקבע . בתוכנית שרצה במחשב הופיע רצף הגישות לכתובות הבאות (hexadecimal) שחזר על עצמו ברציפות 1001 פעמים:

xABCDEF010, xBCDEEF010, xCDEFEF010, xDEF0EF010, xEF01EF010, xF012EF010, x0123EF010, x1234EF010.

בהתעלם מהאיטרציה הראשונה, האם הרצף הנ"ל גורם להחטאות? אם כן, לכמה? (6 נקודות)

מאחר ו- , ה- cache הינו 4-way set associative, כלומר בכל set ישנם 4 בלוקים. אם כך, כמות ה- sets הינה: , ולכן גודל האינדקס לצורך גישה ל- set הינו 11 סיביות. בכל הכתובות הנ"ל כל 14 סיביות ה- LSBs זהות (אלו הסיביות הקובעות את ה- index + offset), ולכן כתובות המלים המתאימות ממופות לאותו set. בתום האיטרציה הראשונה ארבע המלים שבכתובות xEF01EF010, xF012EF010, x0123EF010, x1234EF010 מצויות ב- cache, ומשם ואילך כל פניה לכתובת גורמת ל- miss, ובסה"כ ישנם misses.

1. מהנדסת צעירה בוגרת בר-אילן נתבקשה לפתור את הבעיה והציעה להצמיד ל- cache שבסעיף א' cache קטן נוסף מסוג fully associative, כך ששני הזיכרונות יהיו אקסקלוסיביים, כלומר שמילה תימצא **רק באחד** משניהם, ואם תימצא בקטן, אזי תתבצע החלפה של מילה בין הקטן לגדול, כך שהמילה המבוקשת תעבור מהקטן לגדול, ובמקומה תעבור מילה מסוימת מהגדול לקטן (על-פי מדיניות LRU). המערכת פועלת ע"פ המתואר בציור שלהלן:



עבור רצף הגישות מהסעיף הקודם, מלאו את מספר ה- misses בטבלה הבאה. הסבירו במדויק את הערכים שמילאתם בטבלה. (6 נקודות)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 16 | 8 | 4 | 2 | 1 | Victim cache size [Blocks] |
|  |  |  |  |  | misses |

הוספת victim cache בגודל של בלוק אחד או של שני בלוקים איננה תפתור את בעיית ההחטאות המתוארת בסעיף הקודם. הוספה של victim cache בגודל של 4 בלוקים ומעלה – תאפס את מספר ה- misses עבור רצף הגישות הנ"ל, משום שזיכרון המטמון יחזיק כל הזמן את 8 הבלוקים הנתונים ברצף הגישות.

1. בניסיון לבדוק את יעילות הפתרון בסעיף י' שונה קוד התוכנית כך שהכתובות לגישה לזיכרון הוגרלו באופן אקראי (בהתפלגות אחידה) מתוך 8 הכתובות המופיעות בסעיף ט', 8 פעמים בכל איטרציה, כולל חזרות (ההגרלות הן בלתי תלויות, כל כתובת יכולה להיבחר מספר פעמים באיטרציה, ויכולה גם שלא להיבחר כלל). מלאו את הטבלה והסבירו במדויק את הערכים. (6 נקודות)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 16 | 8 | 4 | 2 | 1 | Victim cache size [Blocks] |
|  |  |  |  |  | misses |

כל פניה אקראית לזיכרון היא בעלת הסתברות של אחידה למאורע שבו הכתובת כבר נמצאת בזיכרון המטמון. לכן, עבור victim cache בגודל בלוק אחד ישנם בזיכרון המטמון (בסט הרלוונטי) בלוקים שונים, והסיכוי להגריל את אחת מתוך 5 הכתובות הללו הוא . לכן, הסיכוי להחטאה הוא: . במהלך 1000 איטרציות ישנן 8000 פניות לזיכרון, ולכן תוחלת ההחטאות תהיה: . ההסבר עבור victim cache בגודל של שני בלוקים דומה.

**שאלה מספר 2 – OoO Execution (55 נקודות)**

נתון מעבד בעל חומרה המבצעת OoO Execution בעזרת אלגוריתם Tomasulo עם ספקולציות (with speculations). נתונים הדברים הבאים:

* רק פקודה אחת מסוגלת לעשות issue בכל מחזור שעון
* ל- ROB ישנן 8 כניסות (slots), והוא משמש גם בתור חוצץ עבור פקודות load ו- store.
* כל היחידות הפונקציונליות (FUs) הינן fully pipelined.
* ישנן שתי FP reservation stations עבור פעולות כפל, ושלוש FP reservation stations עבור פעולות חיבור. כמו-כן, ישנן שלוש integer reservation stations אשר משרתות גם את פקודות ה- load וה- store.
* הנח שלא מתרחשות חריגות (exceptions) במהלך ריצת התוכנית.
* זמן הביצוע (execution) של פעולות integer הוא מחזור שעון אחד. אם מדובר בפקודת load אזי הנח כי באותו מחזור שעון של חישוב הכתובת האפקטיבית מתבצעת גם הפניה והקריאה מהזיכרון. להוציא structural hazards, פקודות load עושות issue במחזור שעון אחד, מבצעות execution במחזור השעון השני, ומבצעות כתיבה (write) במחזור השעון השלישי, כך שפקודה שתלויה בתוצאה של פקודת ה- load יכולה להתחיל את שלב ה- execution שלה במחזור השעון הרביעי.
* זמן הביצוע (execution) של פעולות FP-multiply הינו 4 מחזורי שעון, וזמן הביצוע של פעולות FP-addition הינו 2 מחזורי שעון.
* כאשר ישנו conflict בכתיבה ל- CDB, ניתנת עדיפות (priority) לפקודה המקודמת יותר.
* ביצוע של פקודה התלויה באופרנד יכול להתחיל במחזור השעון לאחר שהאופרנד הדרוש שודר על ה- CDB.
* הנח כי כל ה- RS, ROB,FUs הינן ריקות ופנויות (not busy) בעת תחילת ביצוע התוכנית שלהלן.
* העמודה "value" מתעדכנת בעת שידור הערך על ה- CDB.

על המעבד מריצים את קטע הקוד הבא:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0(R2) | F0, | LD |
| F0 | F0, | F4, | ADDD |
| F2 | F4, | F4, | SUBD |
| F4 | F6, | F0, | MULTD |
| 8 | R2, | R2, | ADDI |
| 1 | R3, | R3, | ADDI |
| F8 | F6, | F2, | ADDD |

הטבלאות הבאות מתארות את מצב החומרה בסוף מחזור השעון האחרון של שלב ה- execution של פקודת ה- ADDD הראשונה (דהיינו, במחזור הבא, שעדיין לא התחיל, פקודת ה- ADDD תכתוב את התוצאה מיחידת הביצוע שלה על ה- CDB).

1. מלאו את הטבלאות כך שיתארו את מצב החומרה בחלוף עוד 5 מחזורי שעון. אם שורה מסוימת עוברת מ- Busy ל- Not Busy, עדכנו את ערך ה- Busy בעמודה המתאימה, אך אל תמחקו את הערכים בשאר העמודות באותה שורה (אלא אם כן פקודה אחרת נכנסת לאותו רגיסטר, ודורסת את הערכים שהיו שם). Integer registers אינם מופיעים בטבלה האחרונה, ואינכם צריכים להתייחס לסטטוס שלהם. (30 נקודות)

|  |
| --- |
| **Reservation** **Station** |
| **Dest** | **Qk** | **Qj** | **Vk** | **Vj** | **Op** | **Busy** | **Name** |
| #2 |  |  | Mem[0+Regs[R2]] | Mem[0+Regs[R2]] | ADDD | Y | Add1 |
| #3 |  | #2 | Regs[F2] |  | SUBD | Y | Add2 |
|  |  |  |  |  |  | N | Add3 |
| #4 | #3 |  |  | Regs[F6] | MULTD | Y | Mult1 |
|  |  |  |  |  |  | N | Mult2 |
| #1 |  |  | 0 | R2 | LD | N | Int1 |
| #5 |  |  | 8 | R2 | ADDI | Y | Int2 |
|  |  |  |  |  |  | N | Int3 |

|  |
| --- |
| **Reservation** **Station** |
| **Dest** | **Qk** | **Qj** | **Vk** | **Vj** | **Op** | **Busy** | **Name** |
| #2 |  |  | Mem[0+Regs[R2]] | Mem[0+Regs[R2]] | ADDD | N | Add1 |
| #3 |  | ~~#2~~ | Regs[F2] | Mem[0+Regs[R2]] × 2 | SUBD | N | Add2 |
| #7 |  |  | Regs[F8] | Regs[F6] | ADDD | N | Add3 |
| #4 | ~~#3~~ |  | Mem[0+Regs[R2]] × 2 - Regs[F2] | Regs[F6] | MULTD | Y | Mult1 |
|  |  |  |  |  |  | N | Mult2 |
| #1 |  |  | 0 | R2 | LD | N | Int1 |
| #5 |  |  | 8 | R2 | ADDI | N | Int2 |
| #6 |  |  | 1 | R3 | ADDI | N | Int3 |

|  |
| --- |
| **Reorder Buffer** |
| **Value** | **Destination** | **State** | **Instruction** | **Busy** | **Entry** |
| Mem[0+Regs[R2]] | F0 | commit | 0(R2) F0, LD | N | 1 |
|  | F4 | Execute | F0 F0, F4, ADDD | Y | 2 |
|  | F4 | Issue | F2 F4, F4, SUBD | Y | 3 |
|  | F0 | Issue | MULTD F0, F6, F4 | Y | 4 |
|  | R2 | Issue | ADDI R2, R2, 8 | Y | 5 |
|  |  |  |  | N | 6 |
|  |  |  |  | N | 7 |
|  |  |  |  | N | 8 |

|  |
| --- |
| **Reorder Buffer** |
| **Value** | **Destination** | **State** | **Instruction** | **Busy** | **Entry** |
| Mem[0+Regs[R2]] | F0 | commit | 0(R2) F0, LD | N | 1 |
| #1 + #1 | F4 | commit | F0 F0, F4, ADDD | N | 2 |
| #2 – Regs[F2] | F4 | commit | F2 F4, F4, SUBD | N | 3 |
| Regs[F6] × #3 | F0 | execute | MULTD F0, F6, F4 | Y | 4 |
| Regs[R2] + 8 | R2 | write result | ADDI R2, R2, 8 | Y | 5 |
| Regs[R3] + 1 | R3 | write result | ADDI R3, R3, 1 | Y | 6 |
| Regs[F6] + Regs[F8] | F2 | write result | ADDD F2, F6, F8 | Y | 7 |
|  |  |  |  | N | 8 |

|  |
| --- |
| **FP Register Status** |
| **F8** | **F7** | **F6** | **F5** | **F4** | **F3** | **F2** | **F1** | **F0** | **Field** |
|  |  |  |  | 3 |  |  |  | 4 | Reorder # |
| N | N | N | N | Y | N | N | N | Y | Busy |

|  |
| --- |
| **FP Register Status** |
| **F8** | **F7** | **F6** | **F5** | **F4** | **F3** | **F2** | **F1** | **F0** | **Field** |
|  |  |  |  | 3 |  | 7 |  | 4 | Reorder # |
| N | N | N | N | N | N | Y | N | Y | Busy |

1. נגדיר את מחזור השעון שבו פקודת ה- LD ביצעה issue. באיזה מחזור שעון תבצע הפקודה האחרונה ברצף הפקודות את שלב ה- commit? (6 נקודות)

פקודת ה- ADDD האחרונה תבצע commit במחזור שעון מספר 18.

1. מהנדס צעיר רצה לבחון את ביצועי המעבד המוזכר בתחילת השאלה לביצועי מעבד אשר מבצע את אלגוריתם Tomasulo, אך איננו תומך בספקולציות (dynamic scheduling, no speculations). מה תהיה ההשפעה על הביצועים אם נריץ את קטע הקוד הנ"ל על מעבד כזה (ללא ROB)? חשבו את ה- speedup. (7 נקודות)

אם קטע הקוד הנ"ל ירוץ על מעבד dynamic scheduling ללא ספקולציות, הרי שפקודת ה- MULTD תבצע write result במחזור שעון מספר 14, כאשר כל שאר הפקודות כבר הסתיימו קודם לכן. מכאן נקבל כי במקרה הזה ישנה האצה בביצועים:

1. הסבירו מהן היתרונות שיש למעבד התומך ב- speculations (כמו זה המתואר בתחילת השאלה) לעומת מעבד שאיננו תומך ב- speculation (ציינו לפחות שני יתרונות בולטים). האם יתרונות אלו באים לידי ביטוי בקטע הקוד הנתון בשאלה? אם כן, הסבירו כיצד. אם לא, הסבירו מדוע. (12 נקודות)

יתרון אחד של מעבד התומך ב- speculation הוא במקרה של פקודות branch אשר אינן מעקבות את ביצוע הפקודות הבאות לאחריהן, וזאת לעומת מעבד dynamic scheduling אשר איננו תומך ב- speculation, שבו כל פקודת branch מחייבת לחכות לשלב ה- branch resolution בטרם נוכל להמשיך ולבצע את הפקודות הבאות לאחר מכן.

יתרון נוסף של מעבד התומך ב- speculation הוא הנושא של התמודדות עם Exceptions (precise exception).

שני היתרונות הנ"ל אינם באים לידי ביטוי במקרה שלנו, היות וקטע הקוד איננו מכיל פקודות branch, וכן היה נתון בשאלה כי ניתן להניח שלא יתרחשו exceptions במהלך ריצת הקוד.