**הטכניון – מכון טכנולוגי לישראל**

**הפקולטה למדעי המחשב**

תאריך הגשה: 5.6.2016 שעה 11:50am

הוראות הגשה: ההגשה בזוגות . הוסיפו שמות, ת.ז., אי-מייל

**להחזיר לתא: 62**

**אילון גל 302921762** [**eilongal@gmail.com**](mailto:eilongal@gmail.com)

**ג'ורג' אפדילא, 312561913,** [**gavdellas@gmail.com**](mailto:gavdellas@gmail.com)

**שאלה 1 – Two level Branch Prediction**

נתון מעבד בעל מבנה ה-pipeline הבא:

Mem

Write

back

Exe

Dec 2

Dec 1

Inst fetch

1. חיזוי הקפיצה נמשך שני מחזורי שעון, כך שהוא מבוצע בסוף שלב dec1 .
2. במעבד קיים branch predictor מסוג Local, שבו לכל branch היסטוריה באורך 3 המצביעה על מערך של 8 מונים (bimodal counters). ההיסטוריה מאותחלת ל-0 והמונים מאותחלים ל-2 (weakly taken) .
3. בסוף שלב ה-mem ידוע האם יש לקפוץ או לא.
   1. כמה penalty (במחזורי שעון) נשלם עבור חיזוי בכל אחד מהמקרים הבאים?

חיזוי taken \ בפועל taken **=>** **מחזור שעון אחד**

חיזוי taken \ בפועל not taken **=> 4 מחזורי שעון**

חיזוי not taken \ בפועל taken **=> 4 מחזורי שעון**

חיזוי not taken \ בפועל not taken **=> 0 מחזורי שעון**

* 1. המעבד מריץ תוכנית ובה פקודת קפיצה בודדת בעלת סידרת הקפיצות הבאה: ...01101 01101 (החל משמאל). השלם את הטבלה הבאה בהנחה שההיסטוריה והמונים מתעדכנים בסוף שלב ה-Mem ושבין קפיצה לקפיצה עובר מספיק זמן, כך שההיסטוריה והמונים מתעדכנים עוד לפני שנדרש לבצע חיזוי נוסף.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Taken/**  **not-taken** | **ערך ההיסטוריה**  **לפני הקפיצה** | **ערך המונה לפני הקפיצה** | **החיזוי**  **(0/1)** | **החיזוי נכון\ שגוי** |
| 0 | 000 | 2 | 1 | שגוי |
| 1 | 000 | 1 | 0 | שגוי |
| 1 | 001 | 2 | 1 | נכון |
| 0 | 011 | 2 | 1 | שגוי |
| 1 | 110 | 2 | 1 | נכון |
| 0 | 101 | 2 | 1 | שגוי |
| 1 | 010 | 2 | 1 | נכון |
| 1 | 101 | 1 | 0 | שגוי |
| 0 | 011 | 1 | 0 | נכון |
| 1 | 110 | 3 | 1 | נכון |

* 1. עבור כל אחת מהסדרות הבאות ענו האם החזאי בשאלה מסוגל לאחר מספיק חזרות לחזות אותן בצורה מושלמת. אם לדעתך החזאי אינו מסוגל לחזות את הדפוס בצורה מושלמת, נמק מדוע, וקבע מה אורך ההיסטוריה **המינימאלי** הנדרש על מנת לתת תשובה נכונה.

א.100010111 100010111 100010111….

**לא ניתן לחזות את הדפוס בצורה מושלמת מכיוון שעבור ההיסטוריה "111", ניתן לראות כי בפעם הראשונה שהרצף הנ"ל מופיע החיזוי נכון ומעביר את המונה לST, ומייד לאחר מכן כשהרצף מופיע בפעם השנייה הקפיצה לא נלקחת בפועל ולכן יש החטאה והמונה עובר למצב WT וכך חוזר חלילה בצורה מחזורית. עבור היסטוריה באורך 4 כל רצף מופיע פעם אחת בלבד ולכן אחרי מספיק חזרות נקבל חיזוי מושלם.**

ב. 0101111 0101111 0101111…..

**לא ניתן לחזות את הדפוס בצורה מושלמת מכיוון שעבור ההיסטוריה "111", ניתן לראות כי בפעם הראשונה שהיסטוריה זו מופיעה החיזוי יהיה נכון והמונה יעבור לST, ומייד לאחר מכן עבור ההיסטוריה "111" אין קפיצה כאשר המונה עדיין על ST ולכן נקבל שגיאה והוא יעבור לWT, ואז ימשיך כך בצורה מחזורית. עבור היסטוריה באורך 4 ניתן לחזות בצורה מושלמת.**

ג. 10110**0**1 10**1**1001 1011001…

**לא ניתן לחזות את הדפוס בצורה מושלמת מכיוון שעבור ההיסטוריה "110" מקבלים לראשונה (סימון ראשון) חיזוי 1 בניגוד ל0 בפועל כך שהמונה יעבור לWNT ובפעם השנייה מקבלים חיזוי של 0 בניגוד ל1 בפועל והמונה עובר לWT. כך זה יקרה בצורה מחזורית כך שלעולם לא נקבל חיזוי מושלם. היסטוריה בגודל 4 לא תספיק במקרה זה כי נקבל חיזוי שגוי עבור ההיסטוריה "0110" בצורה דומה ולכן דרושה היסטוריה באורך 5.**

4.מריצים benchmark בעל מספר פקודות גדול על המעבד. 20% מהפקודות הן פקודות קפיצה. 60% מהחיזויים הם חיזוי Taken . 75% מחיזוי ה-not taken ו- 50% מחיזוי ה- taken נכונים . כל ה- data hazards בקוד נפתרים באמצעות forwarding . אין החטאות במטמון במהלך ריצת התוכנית. מהו ה- cpi של התוכנית?

**נניח שבוחנים תוכנית עם 100 פקודות לפי הפירוט לעיל. נקבל:**

**20 פקודות קפיצה, 12 מתוכם חיזוי taken, 8 מתוכם חיזוי not taken.**

**6 פקודות מתוך החיזוי not taken נכונות, 2 לא נכונות.**

**6 פקודות מתוך חיזוי הtaken נכונות, 6 לא נכונות.**

**לכן מספר המחזורים עבור פקודות הקפיצה בהתחשב בpenalty שחישבנו יהיה:**

**# Branch Cycles = 6 (Not taken, hit) + 2 + 2\*4 (Not taken, miss) + 6 + 6 (Taken, hit) + 6 + 6\*4 (Taken, miss) = 6 + 10 + 12 + 30 = 58.**

**לפי הנתון, מכיוון שכל הhazards נפתרו, נניח כי עבור כל פקודה אחרת מספר המחזורים הוא 1. בסה"כ נקבל:**

**CPI = # of total cycles / # of Instr. = 58 (Branch cycles) + 80 (Non-branch cycles) / 100 (Instr.) = 138/100 = 1.38**

**שאלה 2 – זיכרון וירטואלי**

א.מנה שני יתרונות לכך שהדפים הוירטואלים והפיסיים מיושרים(aligned) יחסית לגודל של הדף (כלומר - עבור גודל דף 4KB , לדוגמא, כתובותיהם יתחילו ב 12 אפסים. )

**1. כפי שלמדנו, אנו בונים על כך ש12 הביטים התחתונים מאופסים על מנת לשמור מידע בתוך הכניסות של טבלאות הדפים (Page Directory & Page Table). לדוגמא, הPTE מצביע למסגרת בזכרון הפיסי, ומכיוון שהוא מיושר 12 הביטים התחתונים מאופסים מה שמאפשר לנו לשמור מידע אודות הדף הזה בתוך הPTE. (אותה תופעה מתאפשרת כאשר יש 2-level ואז הPage Directory מצביע לPage Table שגם הוא מיושר, ואז הכניסה מכילה מידע על הPage Table)**

**2. מכיוון שהדפים הוירטואליים והפיסיים מיושרים אנחנו יכולים להשתמש בoffset בצורה שאנו משתמשים בה: מכיוון שגם הדף הוירטואלי וגם הדף הפיסי מיושרים על גודל של דף ההיסט מתחילת הדף הוירטואלי שווה להיסט מתחילת הדף הפיסי שווה ל12 הביטים התחתונים של הכתובת וכך זה למעשה חוסך תרגום של 12 הביטים, כלומר חסכון ישיר בזמן תרגום ובכמות הנתונים שצריך לשמור כדי לתרגם כתובת וירטואלית לכתובת פיסית.**

ב. בארכיטקטורת x86 עם גודל כתובת של 32 ביט, אילו סוגי טבלאות תרגום חייבים להימצא בזיכרון הראשי?

**עבור 2-level, שבו יש Page Directory שמצביע לטבלאות דפים, הPage Directory חייבת להמצא בזכרון הראשי, מכיוון שהחומרה מתחילה את פענוח הכתובת מטבלה זו, ע"י גישה לכתובת הפיזית המוצבעת ע"י הרגיסטר CR3 ולכן טבלה זו היא חייבת להמצא בזכרון הראשי. הPage Tables לא חייבות להמצא בזכרון הראשי. לאחר הגישה לPGD נקבל כתובת של הPT, ואם היא לא נמצאת בזכרון אז או שהיא לא הוקצתה עדיין וניתן להקצות אותה, או שהיא פונתה בSwapping וצריך לקרוא אותה מהדיסק.**

**תמיד הטבלה הראשונה שממנה מתחיל התרגום צריכה להמצא בזכרון הראשי, כי החומרה משתמשת בכתובת הפיזית שלה כדי להתחיל את התרגום.**

**שאלה 3- זיכרון וירטואלי**

נתון מעבד דמוי x86 העובד במוד של 64 ביט ומבנה הכתובת הבא:

11 12 23 24 31 32 39 40 47 48 63 0

offset

PTE

DIR

PDP

PML4

Sign ext

המעבד תומך הן בדפים בגודל קטן (המוצבעים ע"י כניסות ב-PTE) והן בדפים בגודל גדול (המוצבעים

ע"י כניסות ב-DIR). גודל כל כניסה בטבלאות הדפים בכל אחת מהרמות היא 8 bytes.

במעבד זה, טבלאות התרגום אינן בהכרח בגודל של דף קטן. במעבד קיימים TLB וכן PMH caches עבור כל הרמות PML4,PDP,DIR. בכל אחד מהם 4 כניסות, direct mapped. למעבד אין STLB .

א.מהם הגדלים של דף קטן ודף גדול במעבד?

**מכיוון שהoffset הוא בן 12 ביטים, גודל דף קטן הוא 2^12bits = 4KB.**

**הoffset במקרה של דף גדול הוא הביטים של הoffset של הדף הקטן + הביטים של הPTE. בסך הכל נקבל 24 ביטים, ולכן גודל הדף הגדול הוא 2^24bits = 16MB.**

ב. כמה סיביות אנו שומרים עבור שדה ה-tag ב- TLB וב PMH caches: PML4, PDP וDIR?

**מכיוון שיש 4 כניסות בכל קאש גודל הset יהיה 2 ביטים. כל שאר הביטים של הVPN שייכים לtag. לכן נקבל:**

**TLB: 36-2=34bits for small page, 24-2=22bits for large page**

**PML4: 8-2=6bits, PDP: 16-2=14bits, DIR: 24-2=22bits**

ג. נתונה סדרת פניות לזיכרון וירטואלי (בבסיס הקסדצימאלי) . עבור כל גישה , רשמו האם הייתה החטאה או פגיעה ב- TLB וב-PMH caches . הוסיפו הסבר מתאים. הניחו שמשתמשים בדפים קטנים בלבד, שהטבלאות התרגום כבר קימות בזיכרון ושה- TLB וה- PMH caches ריקים בתחילת התוכנית.

|  |  |  |
| --- | --- | --- |
| הסבר | TLBs(H/M) | כתובות |
| מכיוון שנתון שהטבלאות ריקות נקבל החטאה בכל הרמות | PML4: **M**  PDP: **M**  DIR: **M**  TLB: **M** | FFFF123456789ABC |
| הביטים של הPML4 ושל הPDP נותרו זהים והוכנסו לקאש בפנייה הקודמת. (לכן H) הDIR שונה (הסט שונה) והכתובת אינה בTLB כי הפנייה הקודמת לא נמצאת באותו דף כמו הפנייה הזו – נקבל M. גישה זו תדרוס את הגישה הקודמת בTLB כי הset זהה = 01. | PML4: **H**  PDP: **H**  DIR: **M**  TLB: **M** | FFFF123457789ABC |
| הכתובת המבוקשת נמצאת באותו דף שבוצעה אליו גישה בפנייה הראשונה. בשלושת הקאשים הראשונים הפנייה הראשונה עדיין לא נדרסה ולכן נקבל H. בTLB פנייה זו נדרסה בפנייה בקודמת ולכן נקבל M. | PML4: **H**  PDP: **H**  DIR: **H**  TLB: **M** | FFFF123456789BCD |
| הPML4 זהה לכל הגישות הקודמות ונקבל H, כל שאר הביטים למעט הoffset שונים ולכן נקבל M בכל שאר הקאשים. בDIR הפנייה הזו תדרוס את הפנייה הקודמת כי הset זהה = 10) | PML4: **H**  PDP: **M**  DIR: **M**  TLB: **M** | FFFF123322788BCD |
| שדות הPML4 והPDP לא נדרסו ולכן נקבל H. הDIR נדרס בפנייה הקודמת ולכן נקבל M. הפנייה הזו בוצעה כבר בפנייה השלישית ומכיוון שהיא לא נדרסה מאז בTLB נקבל שם H, לכן שאר הקאשים לא רלוונטים כי קיים תרגום. | PML4: **--**  PDP: **--**  DIR: **--**  TLB: **H** | FFFF123456789BCD |

ד. זמן הגישה לזיכרון הוא 100 מ"ש(מחזורי שעון). זמן הגישה ל- TLBהוא 2 מ"ש (להחטאה או קבלת הנתון). לאחר TLB miss מתבצעת פנייה ל-PMH כדי שיבצע page walk להשגת התרגום. זמן הגישה ב-PMH בכל הרמות הוא 3 מ"ש עד לקבלת הנתון או זיהוי החטאה. למעבד יש זיכרון מטמון : גודלו 32 kb ,גודל שורה 64 בתים וארגון 8-way set associative עם מדיניות פינו LRU .זמן הגישה למטמון (לקבלת נתון או החטאה) הוא 4 מ"ש.

עבור כל אחת מהגישות לזיכרון שהוזכרו בסעיף ב', תוך כמה מ"ש התקבל תרגום לכתובת?

יש לחשב את הזמן הנדרש לקבלת תרגום עבור כל אחת מהכתובות, ולפרט את הגישות השונות שבוצעו לקבלת התרגום, את תוצאת כל גישה (hit או miss) ואת הזמן שהגישה ארכה.

|  |  |  |
| --- | --- | --- |
| הסבר | TLBs(H/M) | כתובות |
| TLB – החטאה = 2 מ"ש, PMH – החטאה = 3 מ"ש,  מטמון מעבד – החטאה = 4 מ"ש עבור כל אחד מהקאשים = 16 מ"ש, גישה לזכרון עבור כל אחד מהקאשים = 400 מ"ש,  **סה"כ: 421 מ"ש** | PML4: **M**  PDP: **M**  DIR: **M**  TLB: **M** | FFFF123456789ABC |
| TLB – החטאה = 2 מ"ש PMH – פגיעה בPML4 ובPDP = 3 מ"ש, מטמון מעבד – פגיעה עבור הDIR (מכיוון שבגישה הקודמת מילאנו cache line שמכיל 8 כניסות של הDIR, כעת נקבל פגיעה) והחטאה עבור הTLB = 4\*2 = 8 מ"ש  גישה לזכרון עבור הTLB = 100 מ"ש  **סה"כ: 113 מ"ש** | PML4: **H**  PDP: **H**  DIR: **M**  TLB: **M** | FFFF123457789ABC |
| TLB – החטאה = 2 מ"ש PMH – פגיעה בPML4, PDP, DIR = 3 מ"ש, מטמון מעבד – פגיעה עבור הTLB והבאת כתובת פיזית = 4 מ"ש  אין גישה לזכרון  **סה"כ: 9 מ"ש** | PML4: **H**  PDP: **H**  DIR: **H**  TLB: **M** | FFFF123456789BCD |
| TLB – החטאה = 2 מ"ש PMH – פגיעה בPML4 והחטאה בשאר = 3 מ"ש, מטמון מעבד – פגיעה עבור הPDP (הובא בגישה הקודמת למטמון) החטאה עבור הDIR והTLB = 4 מ"ש לכל אחד = 12 מ"ש  גישה לזכרון עבור הDIR, TLB = 100\*2 מ"ש = 200 מ"ש  **סה"כ: 217 מ"ש** | PML4: **H**  PDP: **M**  DIR: **M**  TLB: **M** | FFFF123322788BCD |
| TLB –פגיעה והבאת כתובת פיזית = 2 מ"ש  **סה"כ: 2 מ"ש** | PML4: **--**  PDP: **--**  DIR: **--**  TLB: **H** | FFFF123456789BCD |