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, תא אליו יש להחזיר את התרגיל ואת תשובותיכם לתרגיל, הדפיסו והגישו לתא הקורס בקומה 1. עבור הגשות באיחור יש להגיש לתא של יוני.

מבנה מחשבים ספרתיים – תרגיל יבש 3

שם	ת.ז.	מייל
עידן אטיאס	201368958	idan2k3@gmail.com
ליאור בן עמי	201182839	liorb_a@hotmail.com

להחזיר לתא: 39

שאלה 1 – Two level Branch Prediction

נתון מעבד בעל מבנה ה-pipeline הבא:

Inst fetch	Dec 1	Dec 2	Exe	Mem	Write back

- חיזוי הקפיצה נמשך שני מחזורי שעון, כך שהוא מבוצע בסוף שלב dec1 .
- 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).
- 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
   תשובה: מחזור שעון אחד
   חיזוי taken \ בפועל not taken
   תשובה: ארבעה מחזורי שעון
   חיזוי not taken \ בפועל taken
   תשובה: ארבעה מחזורי שעון
   חיזוי not taken \ בפועל not taken
   תשובה: ללא תשלום

2. 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	F
1	000	1	0	F
1	001	2	1	T
0	011	2	1	F
1	110	2	1	T
0	101	2	1	F
1	010	2	1	T
1	101	1	0	F
0	011	1	0	T
1	110	3	1	T

3. 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

<u>תשובה:</u> החזאי אינו יכול לחזות בצורה מושלמת, אורך ההיסטוריה המינימלי עבור סדרה זאת הוא 4. הסבר: נשים לב כי עבור רצף ההיסטוריה 111 יש שני חיזויים ברצף: 1 ו -0. בפעם הראשונה החיזוי הוא WT=2 ולכן יהיה נכון ויעבור ל ST=3. אך מיד לאחר מכן לא תתבצע קפיצה ולכן יהיה שגוי ויעבור ל WT שוב. בפעם השלישית נחזור למקרה הראשון וחוזר חלילה. עבור שאר הרצפים באורך 3 החזאי תקין לכן צריך היסטוריה באורך 4 ומעלה כדי שהחיזוי יהיה תקין תמיד.

ב. 0101111 0101111 0101111

<u>תשובה:</u> החזאי אינו יכול לחזות בצורה מושלמת, אורך ההיסטוריה המינימלי עבור סדרה זאת הוא 4. הסבר: זהה לסעיף הקודם.

ג. ...1011001 1011001 1011001

<u>תשובה:</u> החזאי אינו יכול לחזות בצורה מושלמת, אורך ההיסטוריה המינימלי עבור סדרה זאת הוא 5. הסבר: עבור הרצף 110 בפעם הראשונה הוא החיזוי אמור לתת 0 אך יתן 1 WT=2, ואז יעבור ל WNT=1. בפעם הבאה ייתן לכן 0 אך אמור לתת 1 וחוזר חלילה. עבור היסטוריה באורך 4, לא יסתדר הפעם מכיוון שהרצף 0110 בעל אותו מבנה (0 ואז 1). הפעם צריך היסטוריה באורך 5 ומעלה כדי שהחיזוי יהיה מושלם.

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?

<u>תשובה</u>

$$CPI_{new} = CPI_{ideal} + \frac{avg\ stall\ cycles}{instr.}$$
$$= CPI_{ideal} + True\ taken + False\ taken + False\ not\ taken$$
$$= 1 + 0.2 \cdot 0.6 \cdot 0.5 \cdot 1 + 0.2 \cdot 0.6 \cdot 0.5 \cdot 4 + 0.2 \cdot 0.4 \cdot 0.25 \cdot 4 = 1.38$$

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.)

תשובה:

1. מנגנון התרגום מכתובת וירטואלית לפיסית מתרגם PPN ➔ VPN ולכן עבור <u>כל</u> כתובת וירטואלית, שכאמור מיושרת לפי גודל דף, זמן התרגום מתקצר כיוון שאין צורך לתרגם (|page|)log ביטים תחתונים.
2. נוכל למקבל גישה ל-Cache ול- TLB:

מכיוון ש- (|page|)log ביטים תחתונים (המבטאים את ה-offset בתוך הדף המבוקש) אינם צריכים תרגום נוכל לחלץ בעזרתם את מספר הסט המתאים ב Cache וע"י כך לקבל את ה – tag המתאים של ה – cache line. בו בזמן, נוכל לקחת את שאר הביטים (העליונים) של הכתובת ולחפש מיפוי של PPN ➔ VPN ב-TLB. במידה וקיבלנו TLB HIT אזי כל שנותר לעשות הוא להשוות האם ה-tag של ה PPN שקיבלנו שווה ל tag של ה cache line ואם כן אז קיבלנו cache hit.

*נשים לב שנוכל לעשות זאת רק כיוון שידוע לנו כי מספר מסוים של ביטים תחתונים לא צריך תרגום. (וזה כאמור נובע מכך שכל הדפים מיושרים יחסית לגודל דף)

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

תשובה:

טבלאות ה-PGD של כל תהליך חייבות להימצא ב-RAM.

טבלת הPGD היא הטבלה הראשונה בהיררכית טבלאות התרגום בארכיטקטורה הנ"ל. ישנו רגיסטר CR3 שתמיד מצביע על הPGD של התהליך הנוכחי וההגדרה היא שתמיד הטבלה (PGD) שמוצבעת ע"י כתובת זו - מוקצית.

שאלה 3- זיכרון וירטואלי (40 נק')

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

63 48	47 40	39 32	31 24	23 12	11 0
Sign ext	PML4	PDP	DIR	PTE	offset

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. (5 נק') מהם הגדלים של דף קטן ודף גדול במעבד?

גודל דף קטן הינו $2^{12} = 4{,}096$ בתים (PTE) במונחים של זיכרון מדובר ב4 קילו בייט.

גודל דף גדול הינו $2^{24} = 16{,}777{,}216$ בתים (DIR) במונחים של זיכרון מדובר ב16 מגה בייט.

ב. (5 נק') כמה סיביות אנו שומרים עבור שדה ה-tag ב- TLB וב PMH caches: PML4, PDP וDIR?

TLB – 34

PML – 6

PDP – 14

DIR – 22

לכל אחד ניקח את מספר הסיביות החל מ sign ext עד סופו פחות 2

ג. (10 נק') 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
נתון שהכל ריק. החטאה ב-TLB.	miss: all	FFFF123456789ABC
נשים לב כי FFFF1234XXXXXXXX ללא התייחסות ל-Xים כבר היינו, ולכן PML4 ו – PDF נקבל פגיעה, נראה שאמנם לאחר מכן ב56 היינו אבל הכתובת שונתה ל57 ולכן החל מנקודה זו נקבל החטאה בכל השאר (PTE & DIR). החטאה ב-TLB, הכתובת לא נמצאת.	hit: PML4 & PDP miss: DIR & PTE	FFFF123457789ABC
כתובת שכבר היינו בה חוץ מחוצץ offset (12 ביטים אחרונים) שהשתנה ל-BCD. החטאה ב-TLB, הכתובת נדרסת.	hit: PML4 & PDP & DIR & PTE	FFFF123456789BCD
נשים לב כי FFFF12XXXXXXXXXX ללא התייחסות ל-Xים כבר היינו, ולכן PML4 נקבל פגיעה, נראה שאמנם לאחר מכן ב34 היינו אבל הכתובת שונתה ל33 ולכן החל מנקודה זו נקבל החטאה בכל השאר (PTE & PDP & DIR). החטאה ב-TLB.	hit: PML4 miss: PDP & DIR & PTE	FFFF123322788BCD
כתובת שכבר היינו בה. פגיעה ב-TLB.	hit: PML4 & PDP & DIR & PTE	FFFF123456789BCD

ד. (20 נק') זמן הגישה לזיכרון הוא 100 מ"ש(מחזורי שעון). זמן הגישה ל-TLB הוא 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	מספר cycles	כתובות
החטאה בכל הרמות.	421	FFFF123456789ABC
גישה לtlb = 2, גישה בpmh = 3 ולאחר מכן גישה לזיכרון = 100, בנוסף יש לנו שתי פגיעות ולכן = 2*4.	113	FFFF123457789ABC
גישה לtlb =2 , בנוסף יש לנו גישה למטמון = 4, וגישה לpmh = 3.	9	FFFF123456789BCD
גישה לtlb = 2, יש לנו גישה לpmh עם פגיעה = 3 , ועוד 2 החטאות ולכן ניגש פעמיים לזיכרון = 2*100, בנוסף 3 גישות למטמון = 3*4.	217	FFFF123322788BCD
פגיעה ב-TLB וסיום.	2	FFFF123456789BCD