ת"ב 2 – חזאי סיעוף Branch Predictor

קצר ולעניין

בתרגיל זה תממשו מדמה חזאי סיעוף בעל שתי רמות (2-Level Branch Predictor), בדומה לנלמד בכיתה. תצורת החזאי תהיה גמישה ותוגדר בתחילת הריצה של המדמה באמצעות פרמטרים. סביבת המדמה תספק trace (עקבות ריצה) מריצת תכנית כלשהי, שיתאר את אירועי הסיעוף בלבד – כתובת פקודה, הכרעת הקפיצה וכתובת יעד מחושבת. המדמה יידרש לתת חיזוי לכל אירוע סיעוף (על פי כתובת הפקודה בלבד) ולאחר מכן לעדכן את מצבו בהתאם לזיהוי הפקודה בפועל והכרעת הסיעוף (תוצאת שלב ה-EXE) בפועל, כפי שמפורט ב-trace. הפרמטרים וה-trace מסופקים בקובץ קלט שנקרא על ידי סביבת המדמה שמסופקת לכם.

מאפייני החזאי

תצורת החזאי, כלומר המאפיינים שניתן לקבוע באמצעות פרמטרים לאיתחול, כוללים:

- גודל טבלת ה-BTB: מספר כניסות בטבלה. ערכים אפשריים: 1,2,4,8,16,32 (חזקות 2 עד 32)
- גודל רגיסטר ההיסטוריה: מספר ביטים. ערכים אפשריים: 1 עד 8
- היסטוריה לוקלית או גלובלית
- טבלאות מכונות מצבים לוקליות או גלובליות
- שימוש ב-Lshare/Gshare – כן או לא. רלוונטי, כמובן, רק כשטבלת מכונות המצבים היא גלובלית.

מאפיינים נוספים (קבועים):

- מכונות המצבים בטבלאות הן Bimodal (2-ביטים) כפי שנלמדו בכתה. מצב התחלתי WNT.
- גודל כתובת במעבד הינו 32 ביט. כל פקודה מתחילה בכתובת מיושרת ל-4.
- ה-XOR במקרה של G/L-share יהיה עם הביטים של כתובת הפקודה החל מביט 2 (השלישי).
- גודל ה-tag בטבלה מאפשר זיהוי חד משמעי של פקודת קפיצה (כלומר מספיק ביטים כדי להבחין בין פקודות בכתובות שונות).

מבנה קובץ הקלט

תצורת החזאי וה-trace נקראים על ידי ה-main שמסופק לכם מתוך קובץ קלט ששמו ניתן בשורת הפקודה של bp_main. מסופקות לכם מספר דוגמאות לקבצי קלט עם חומרי התרגיל.

השורה הראשונה בקובץ הקלט מכילה רצף שדות של מאפייני החזאי עם רווח בין כל אחד מהם:

1. מספר הכניסות ב-BTB.
2. מספר הביטים בהיסטוריה.
3. local_history או global_history
4. local_tables או global_tables
5. using_share או not_using_share

לדוגמה:

```
16 5 global_history global_tables using_share
```

כל השורות הבאות בקובץ מכילות עקבות (trace) של פקודות סיעוף מריצת תוכנית כלשהי, כאשר כל שורה תכיל תיאור של עיבוד פקודת סיעוף אחת במבנה של 3 שדות עם רווח ביניהם:

1. כתובת פקודת הסיעוף.
2. הכרעת הסיעוף: T (Taken) או N (Not taken).
3. כתובת יעד קפיצה מחושב (גם אם Not taken).

לדוגמה:

```
0x1036 N 0x1050
```

אז מה תכל'ס צריכים לעשות?

אנחנו מספקים לכם קובץ bp_main.c שכבר מבצע קריאה של קובץ הקלט כמפורט לעיל. ה-main קורא לפונקציות החזאי שלכם על מנת לאתחל אותו עם התצורה שנקראה ולאחר מכן קורא לפונקציות לחיזוי ועדכון מצב החזאי, שאותן עליכם לממש.

עליכם לממש את החזאי בקובץ bp.c או bp.cpp . על החזאי לממש את הפונקציות המוגדרות בקובץ bp_api.h:

1. פונקציית איתחול החזאי. מגדירה את תצורת החזאי לפני תחילת עבודה.

```
int BP_init(unsigned btbSize, unsigned historySize,
            bool isGlobalHist, bool isGlobalTable, bool isShare);
```

2. פונקציית חיזוי לשימוש בשלב כמו IF. מקבלת ערך PC של פקודה נוכחית ומחזירה חיזוי – true עבור Taken. במקרה של חיזוי קפיצה שנלקחת, בפרמטר dst יוחזר ערך PC של יעד הקפיצה המחושב. אם PC של פקודה לא מוכר ב-BTB אזי יש להחזיר false (כלומר, מבחינת החזאי זו אינה פקודת קפיצה, נכון לרגע זה) ויעד הקפיצה יהיה pc+4.

```
bool BP_predict(uint32_t pc, uint32_t *dst);
```

3. פונקציית זיהוי פקודת סיעוף. לשימוש בשלב ID, כאשר פקודה זוהתה כפקודת סיעוף בפועל. מיידע את החזאי שיש פקודת סיעוף בכתובת הנתונה. אם הכתובת לא ידועה, החזאי צריך להכניס ל-BTB רשומה חדשה עבורה, אחרת (הכתובת מוכרת בחזאי) הקריאה לפונקציה זו לא משנה דבר.

```
void BP_setBranchAt(uint32_t pc);
```

4. פונקציית עדכון מצב חזאי עבור פקודת סיעוף. משמש לאחר שלב ה-EXE לעדכון מצב החזאי בהתאם להכרעת פקודת הסיעוף.

```
void BP_update(uint32_t pc, uint32_t targetPc, bool taken);
```

שימו לב ש-`targetPc` הוא הכתובת המחושבת לקפיצה, גם אם היא not taken (מה שיישמר ב-BTB).

שימו-לב: ה-main שניתן נועד להקל עליכם בבדיקה, אולם אתם מחויבים למימוש הממשק לחזאי כפי שמוגדר ב-bp_api.h. כלומר, ייתכן והחזאי ייבדק בדרכים שונות מהמודגם ב-main. למשל, ניתן לקרוא לפונקציית החיזוי עם פקודה שאיננה כלל מסוג Branch ולכן לא תקרא עבורה הפו' setBranchAt ו- update. לכן, הקפידו לעמוד בדרישות הממשק כפי שמתועדות בקובץ bp_api.h.

ארגון ה-BTB

טבלת ה-BTB מקצה כניסה לכל פקודת סיעוף שנתקלים בה במהלך ריצה (עד כדי מגבלת מקום). על מנת לייעל חיפוש בטבלה, לעיתים קובעים את מספר הכניסה בטבלה עבור פקודה מסוימת באופן עקבי, על ידי שימוש במספר ביטים מכתובת הפקודה בזיכרון. לדוגמה, אם נגדיר את ה-BTB בגודל 16 כניסות, כלומר 2^4, אזי נוכל להשתמש בארבע הביטים [2:5] של ה-PC של פקודת סיעוף על מנת לבחור את הכניסה בטבלת ה-BTB בה נשים את אותה פקודה. באופן הזה, החזאי יצטרך לחפש פקודת סיעוף רק בכניסה מסוימת – פעולה יעילה יותר מאשר חיפוש בכל הטבלה. בכל כניסה יהיה TAG , כפי שנלמד בכתה, על מנת לזהות את הכתובת המלאה של הפקודה שנשמרה שם.

לשיטת הקצאה כזו של כניסות בטבלה נקרא direct mapping. מהגדרת אופן המיפוי של PC לכניסה בטבלה, ברור כי ייתכן ותהיה התנגשות בטבלה בין שתי פקודות סיעוף בעלות אותו ערך בביטים המשמשים למיפוי (ביטים 2-5 בדוגמה). במצב כזה, פעולת הכנסת פקודת סיעוף אחת תחליף את הרשומה של הפקודה השנייה שכרגע בטבלה באותה הכניסה. כלומר, בטבלה תיוותר רק הפקודה האחרונה שטופלה מבין השתיים שמתמפות לאותה הכניסה בטבלה.

בתרגיל זה אתם מתבקשים לממש את ההקצאות של הכניסות ב-BTB בשיטת direct mapping הנ"ל, תוך שימוש בביטים של ה-PC, החל מביט 2 (השלישי), בהתאם לגודל טבלת BTB שהוגדר.

שימו לב שבכל פעם שמוכנסת פקודת סיעוף חדשה בכניסה כלשהי ב-BTB - לא משנה אם לכניסה ריקה או במסגרת החלפה - נאפס את רגיסטר ההיסטוריה של אותה כניסה (במקרה של היסטוריה לוקלית) וכן נאתחל את טבלת מכונות המצבים המקושרת לכניסה למצב ברירת המחדל WNT (במקרה של טבלאות לוקליות, בלבד).

לדוגמה:

BTB (size 4)		
tag	target	Local history (size 4)
0x100	0xAA230	0000
0x104	0xB340	1000
0x108	0xC450	0110
0x10C	0xDD670	0101

למשל, פקודת ה-Branch בכתובת 0x108 שביטים [2:3] של הכתובת שלה הם 10 נכנסת למקום השלישי ב-BTB המתאים לביטים 10.

במידה וכעת מגיעה פקודת Branch שכתובתה 0x128 היא נכנסת לאותו המקום ב-BTB (במקום ה-Branch בכתובת 0x108).

דרישות ההגשה

הגשה אלקטרונית בלבד באתר הקורס ("מודל") מחשבונו של אחד הסטודנטים.

מועד ההגשה: עד יום א' 08.05.2016 בשעה 23:55.

עליכם להגיש קובץ ZIP בשם hw2_*ID1_ID2*.zip כאשר ID1 ו-ID2 הם מספרי ת.ז. של המגישים. לדוגמה: hw2_012345678_987654321.zip . ה-ZIP יכיל קובץ **יחיד**:

- קוד המקור של החזאי שלכם: bp.c או bp.cpp .
  קוד המקור חייב להכיל תיעוד פנימי במידה סבירה על מנת להבינו.

דגשים להגשה:

1. המימוש שלכם **חייב** להתקמפל בהצלחה ולרוץ בסביבת השרת הטכניוני T2 (או TX). יש להשתמש בקומפיילר G++/GCC ברירת המחדל בשרת ללא שום דגלים מיוחדים מעבר למה שמשתמשים ב-makefile המסופק לכם.
2. שימו-לב לסנקציות במקרה איחור כמפורט באתר הקורס. מאוד לא מומלץ לאחר את מועד ההגשה שהוגדר לתרגיל. בנסיבות מיוחדות יש לקבל אישור **מראש** מהמתרגל האחראי לאיחור.
3. מניסיונם של סטודנטים אחרים: הקפידו לוודא שהקובץ שהעלתם ל"מודל" הוא אכן הגרסה שהתכוונתם להגיש. לא יתקבלו הגשות נוספות לאחר מועד ההגשה שנקבע בטענות כמו "משום מה הקובץ במודל לא עדכני ויש לנו גרסה עדכנית יותר שלא נקלטה".

בהצלחה!