אוניברסיטת בר-אילן (ע"ר)
הפקולטה להנדסה
רמת-גן 52900

BAR-ILAN UNIVERSITY (RA)
Faculty of Engineering
Ramat-Gan 52900, Israel

Tel: 03-5317722

engbi@mail.biu.ac.il

מבנה מחשבים ספרתיים

תשפ"א סמסטר ב' מועד ב'

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**: כל חומר עזר מודפס ומחשבון.
- **משך הבחינה:** שלוש שעות.
- **יש לצרף את שאלוני הבחינה למחברת!**

בהצלחה!

<u>שאלה מספר 1 – Multithreading (60 נקודות)</u>

בשאלה זו נבחן ביצועים של מעבדי multithreaded.

נניח שני מעבדים אפשריים:

1) מעבד in-order אשר הינו superscalar ברוחב 2. המעבד מרובה חוטים (multithread) במדיניות fine-grain במצב round-robin. המעבד יכול להוציא שתי פקודות מכל סוג (ALU, MEM, FPU וכו') בכל מחזור שעון (גם אם מדובר בשתי פקודות מאותו סוג).
2) מעבד Out-of-Order אשר הינו superscalar ברוחב 2. המעבד מרובה חוטים (multithread) במדיניות fine-grain במצב round-robin, ויש לו ROB בגודל 100 כניסות.

- שני המעבדים תומכים ב- forwarding מלא, לכן, פקודה התלויה בערך מסוים יכולה להתבצע מיד במחזור השעון העוקב לזה שבו חושב הערך שבו היא תלויה.
- כל אחד מהמעבדים יכול להוציא שתי פקודות מכל סוג (ALU, MEM, FPU וכו') בכל מחזור שעון (גם אם מדובר בשתי פקודות מאותו סוג).
- במקרה שהמעבד מריץ יותר מחוט יחיד, שלב ה- Fetch יביא במחזור מסוים עד שתי פקודות מחוט אחד, ובמחזור השעון העוקב יביא עד שתי פקודות מחוט אחר. בכל מקרה, לא יובאו פקודות מחוטים שונים באותו מחזור שעון.
- ה- ROB (במעבד השני) מפוצל באופן שווה בין החוטים (לדוגמא, במצב של שני חוטים כל אחד יקבל 50 כניסות ב- ROB).

מערכת הזיכרון של כל מעבד:

- מטמון L1 בגודל של 32KB, 4-way set-associative, זמן גישה של מחזור יחיד, גודל בלוק 64 בתים, מדיניות כתיבה write-allocate.
- מטמון L2 בגודל של 512KB, 4-way set-associative, זמן גישה של 20 מחזורי שעון, גודל בלוק 64 בתים, מדיניות כתיבה write-allocate.
- זמן הגישה לזיכרון הראשי הינו 400 מחזורי שעון.

מריצים על כל חוט בנפרד את הפונקציה המתוארת להלן, המפצלת הכפלת מטריצות בגודל N*N למספר חוטים. הפרמטר threadID הוא האינדקס של החוט הספציפי, והפרמטר numThreads הוא מספר החוטים הכולל בהם המעבד תומך.

```
matMul (float* src1, float* src2, float* dst, int threadID, int numThreads)
{
        for (int i= threadID; i < N*N; i += numThreads) {
                int row = i / N;  // integer division
                int col = i % N;  // modulo operation
                for (int k=0; k<N; k++) {
                        dst[row * N + col] += src1[row * N + k] * src2[k * N + col];
                }
        }
}
```

הלולאה הפנימית (המודגשת) מתורגמת ל- 6 פקודות מכונה:

1	Loop:	LD	R1 ← Mem[row*N + R4]
2		LD	R2 ← Mem[row*N + R4]
3		FMA	R3 ← (R3 + R1*R2)
4		ADDI	R4 ← R4 + 1
5		SD	Mem[row*N + col] ← R3
6		BNE	R4, N, Loop

הנחות:

- שני המעבדים מחשבים קריאה מהזיכרון או כתיבה לזיכרון בצורת: Rd=Mem[Ra*Rb+Rc] או בצורת: Mem[Ra*Rb+Rc]=Rd בהתאמה, בפקודה אחת שזמן ביצועה הוא מחזור שעון יחיד, וזאת בנוסף לזמן הבאת המידע מהזיכרון.
- יחידות הזיכרון בשני המעבדים מסוגלות להריץ פקודות חדשות גם כאשר הן מחכות לתשובות (למידע) מהזיכרון.
- פקודת FMA הינה פקודת fused multiply-add אשר מבצעת כפל וחיבור מהצורה: Rd=Ra*Rb+Rc
- זמן ביצוע של כל פקודה חישובית (ALU, FPU) או קפיצה מותנית הוא מחזור שעון יחיד.
- למעבדים חזאי קפיצות מושלם.
- גודל משתנה מסוג float הוא 4 בתים.
- המחסניות של החוטים לא תופסות מקום בזיכרון הראשי או בזיכרונות המטמון.
- לצורך חישובי ה- IPC ניתן להזניח את זמן מילוי הצינור בתחילת הריצה.
- בתחילת הריצה שלוש המטריצות כבר נמצאות במטמון/בזיכרון הקטן ביותר שיכול להכיל אותן (אין החטאות מסוג compulsory).
- ניתן להניח כי למעבד השני (OoO) ישנן מספיק reservation stations.

א. (10 נקודות) רשום את חלוקת הכתובת לשדות (tag/index/offset) עבור גישה ל- L1 ועבור גישה ל- L2.

לגבי הגישה ל- L1:

גודל בלוק הינו $64\text{B} = 2^6\text{B}$, ולכן מספר הסיביות בשדה ה- offset הינו 6 סיביות [5:0].

גודלו של L1 הינו $32\text{KB} = 2^{15}\text{B}$, ולכן מספר הבלוקים ב- L1 הינו:

$$\#blocks = \frac{|cache|}{|block|} = \frac{32\text{KB}}{64\text{B}} = 2^{15-6} = 2^9$$

היות ו- L1 הינו 4-way set-associative, לכן, מספר ה- sets ב- L1 הוא:

$$\#sets = \frac{|cache|}{|block| \cdot 4_{way}} = \frac{32\text{KB}}{64\text{B} \cdot 4} = 2^7$$

לכן, יש צורך ב- 7 סיביות בשביל שדה ה- index [12:6].

במהלך הבחינה נאמר לנבחנים כי אורך כתובת הוא 32 סיביות, ולכן:

מספר הסיביות בשדה ה- tag הוא: $32 - 7 - 6 = 19$. שהן סיביות [31:13].

לגבי הגישה ל- L2:

שדה ה- offset זהה (6 סיביות), משום שגודל הבלוק זהה.

גודלו של L2 הינו $512\text{KB} = 2^{19}B$, ולכן מספר הבלוקים ב- L2 הינו:

$$\#blocks = \frac{|cache|}{|block|} = \frac{512\text{KB}}{64\text{B}} = 2^{19-6} = 2^{13}$$

היות ו- L2 הינו 4-way set-associative, לכן, מספר ה- sets ב- L2 הוא:

$$\#sets = \frac{|cache|}{|block| \cdot 4_{way}} = \frac{512\text{KB}}{64\text{B} \cdot 4} = 2^{11}$$

לכן, יש צורך ב- 11 סיביות בשביל שדה ה- index [16:6].

מספר הסיביות בשדה ה- tag הוא: $32 - 11 - 6 = 15$. שהן סיביות [31:17].

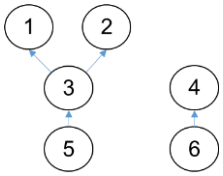
ב. (10 נקודות) נרצה להשתמש במעבד הראשון (המעבד in-order) עבור גודל מטריצה N=150. מהו מספר החוטים המינימלי בהם המעבד צריך לתמוך כדי להבטיח ביצועים מקסימליים, דהיינו IPC= 2 ? נמקו.

ישנן 3 מטריצות בגודל N*N כל אחת (dst, src1, src2), כל תא הוא float, גודל כולן ביחד:

$$3 \cdot \text{N} \cdot |float| = 3 \cdot 150^2 \cdot 4\text{B} = 3 \cdot 300^2 \cdot 1\text{B} = 270{,}000\text{B}$$

כלומר, כל המטריצות נמצאות ב- L2. צריך לחפות על 21 מחזורי שעון נוספים של פעולות load/store (במקרה של L1 miss), וזה צריך להתבצע בעזרת N-1 חוטים, היות והחוט שביצע את פעולת ה- load/store איננו משתתף במשימה. לכן, צריך להתקיים: $N - 1 = 21$, ולכן יש צורך ב- 22 חוטים.

צריך גם לוודא שבאמת אפשר לבצע במקביל כל שתי פקודות עוקבות.



גרף התלויות (תלויות מידע אמתיות, דהיינו: RAW) של התוכנית הוא:

ניתן לראות כי אכן אפשר לבצע במקביל כל שתי פקודות עוקבות, ולכן מספיק 22 חוטים.

ג. (10 נקודות) שוב נשתמש במעבד הראשון, אך הפעם גודל המטריצה הינו N=500. מהו מספר החוטים המינימלי בהם המעבד צריך לתמוך כדי להבטיח ביצועים מקסימליים, דהיינו IPC= 2 ? האם הדבר מעשי מבחינה חומרתית? נמקו תשובתכם.

ישנן 3 מטריצות בגודל N*N כל אחת (dst, src1, src2), כל תא הוא float, גודל כולן ביחד:

$$3 \cdot \text{N} \cdot |float| = 3 \cdot 500^2 \cdot 4\text{B} = 3 \cdot 1000^2 \cdot 1\text{B} = 3 \cdot 10^6\text{B}$$

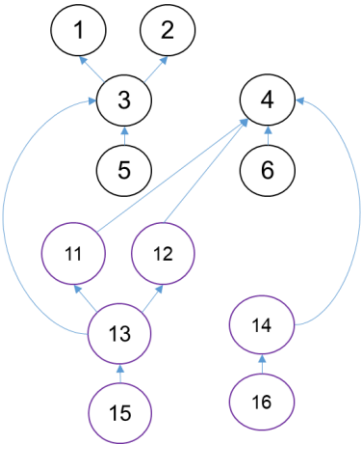
כלומר, כל המטריצות נמצאות בזיכרון הראשי. צריך לחפות על 421 מחזורי שעון נוספים של פעולות load/store (במקרה של L1+L2 miss), וזה צריך להתבצע בעזרת N-1 חוטים. לכן, צריך להתקיים: $N - 1 = 421$, ולכן יש צורך ב- 422 חוטים (כבר ראינו בסעיף הקודם כי ניתן להריץ כל 2 פקודות עוקבות במקביל). הדבר לא מעשי מבחינה חומרתית, משום שעבור כל חוט יש צורך ב- Register-File, PC, Page-Table וכו'.

ד. (10 נקודות) כעת, נרצה להשתמש במעבד השני (מעבד Out-of-Order) עבור גודל מטריצה N=40. מהו מספר החוטים המינימלי בהם המעבד צריך לתמוך כדי להבטיח ביצועים מקסימליים, דהיינו IPC= 2 ? תשובה ללא נימוק לא תתקבל.

ישנן 3 מטריצות בגודל N*N כל אחת (dst, src1, src2), כל תא הוא float, גודל כולן ביחד:

$$3 \cdot \text{N} \cdot |float| = 3 \cdot 40^2 \cdot 4\text{B} = 3 \cdot 80^2 \cdot 1\text{B} = 19{,}200\text{B}$$

כלומר, כל המטריצות נמצאות ב- L1. צריך לחפות על מחזור שעון אחד נוסף של פעולות load/store. בפתרון זה (הן בגרפים והן בטבלאות בסוף המסמך) נציין כל פקודה במספר, כאשר ספרת האחדות מציינת את מספר הפקודה בתוך האיטרציה, וספרת העשרות מציינת את מספר האיטרציה. לדוגמא: פקודה המסומנת במספר 12 הינה פקודה מספר 2 באיטרציה מספר 1, באופן דומה: פקודה המסומנת במספר 31 הינה פקודה מספר 1 באיטרציה מספר 3.



גרף התלויות (RAW) בין שתי איטרציות עוקבות הוא:

נשים לב, כי פקודות ה- Load של כל איטרציה הן בלתי תלויות, ולכן, בחוט אחד יש מספיק אי-תלות בין איטרציות כדי להחביא את הגישות לזיכרון של מחזור שעון יחיד. צריך רק לוודא שיש מספיק מקום ב- ROB.

ניתן לראות בטבלה המצורפת בסוף המסמך כי המחזוריות של ה- OoO מבחינת ביצוע פקודות היא מחזוריות קצרה, ולא מהווה בעיה מבחינת גודל ה- ROB. לכן, די בחוט בודד על מנת להבטיח ביצועים מקסימליים.

ה. (10 נקודות) שוב נשתמש במעבד השני, אך הפעם גודל המטריצה הינו N=150. מהו מספר החוטים המינימלי בהם המעבד צריך לתמוך כדי להבטיח ביצועים מקסימליים, דהיינו IPC= 2 ? האם זה בכלל אפשרי במעבד הנתון? תשובה ללא נימוק לא תתקבל.

מסעיף ב' אנחנו כבר יודעים כי כל המטריצות נמצאות ב- L2. צריך לחפות על 21 מחזורי שעון נוספים של פעולות load/store. נרצה להסתפק בחוט בודד, אך צריך לבדוק האם ה- ROB מספיק גדול כדי להכיל כמות פקודות שימסכו את הגישה הארוכה לזיכרון. במהלך 21 מחזורי שעון המעבד יבצע fetch ל- 42 פקודות, וכמובן שיהיו עוד מספר קטן של פקודות ב- ROB (תלוי במחזור השעון). מאחר וגודל ה- ROB הוא 100 ← אין שום בעיה למסך את הגישה לזיכרון. ולכן, גם הפעם נוכל להגיע לביצועים הנדרשים עם חוט בודד.

ניתוח יותר מפורט של הסעיף (לא נדרש בבחינה): פקודות 3,5 מכל איטרציה יעוכבו בגלל הגישות לזיכרון, אבל פקודות 1,2,4,6 יוכלו להתבצע. פקודות 3,5 יעוכבו 21 מחזורי שעון, משום שהן תלויות בפקודות 1,2 של אותה איטרציה. איטרציה נכנסת ל- ROB ב-3 מחזורי שעון (כל איטרציה היא 6 פקודות, ורוחב ה- IF וה- ID הוא 2). לכן, בחלון הזמן של 21 מחזורי שעון יכנסו ל- ROB 7 איטרציות. האיטרציות הללו לא יוכלו לעשות commit משום שזה נעשה in-order. לכן, בזמן ה- commit של פקודות 3,4 של איטרציה מסוימת יהיו 7 איטרציות ב- ROB, ואז נתחיל לעשות commit לכל הפקודות, ולכן יישארו ב- ROB 7 איטרציות וקצת בכל רגע נתון. 7 איטרציות הן 42 פקודות, ולכן גודל ה- ROB מספיק להשגת ביצועים של $IPC = 2$.

לשם הבנה, ניתן להיעזר בטבלה המצורפת בסוף הקובץ, שם ניתן לראות כיצד באיטרציות הראשונות הפקודות נאספות ב- RS וב- ROB. אולם, בשלב מסוים פקודות 3 ו- 5 של האיטרציות יוכלו להתבצע (מחזור 26 לאיטצריה הראשונה ומחזור 29 לאיטרציה השנייה). בשלב זה כבר נצליח להריץ 2 פקודות בכל מחזור שעון (ב-3 מחזורי שעון נבצע 4 פקודות מהאיטרציה הנכנסת + פקודות 3,4 מאיטרציות קודמות).

ו. (10 נקודות) גם בסעיף זה נשתמש במעבד השני, הפעם עבור גודל מטריצה של N=500. מהו מספר החוטים המינימלי בהם המעבד צריך לתמוך כדי להבטיח ביצועים מקסימליים, דהיינו IPC= 2 ? האם זה בכלל אפשרי במעבד הנתון? תשובה ללא נימוק לא תתקבל.

מסעיף ב' אנחנו כבר יודעים כי כל המטריצות נמצאות בזיכרון הראשי. צריך לחפות על 421 מחזורי שעון נוספים של פקודות load/store. גם אם ה- ROB מלא בפקודות מוכנות לביצוע (מחוט יחיד או מהרבה חוטים), הם יתבצעו לכל היותר ב- 50 מחזורי שעון (שתי פקודות בכל מחזור שעון, וגודל ה- ROB הוא 100 כניסות). זה לא מספיק כדי לחפות על גישה יחידה לזיכרון הראשי, ולכן המעבד יצטרך להתעכב. לכן, לא ניתן להגיע ל- $IPC = 2$ במקרה הזה.

<u>שאלה מספר 2 – Branch Prediction (60 נקודות)</u>

נתון מעבד in-order עם צינור יחיד בעל חמישה שלבים (IF, ID, EX, MEM, WB).

הכרעת הקפיצות (כולל חישוב כתובת היעד) נמצאת בשלב ה- MEM.

למעבד קיים 2-level branch predictor עם הפרמטרים הבאים:

- 4 כניסות, fully-associative עם מדיניות החלפה LRU.
- היסטוריה לוקאלית באורך 2 סיביות.
- מכונות מצבים פרטיות מסוג 2-bit saturating counter המאותחלות ל: WNT=01
- כל פקודת branch שנכנסת לחזאי מתחילה ממצב מאופס (היסטוריה שכולה אפסים, כל מכונות המצבים מאותחלים ל- WNT)
- החיזוי נעשה בשלב ה- IF.
- אורך כתובת הוא 32 סיביות.
- אורך פקודה הינו 4 בתים, וכל הפקודות מיושרות בזיכרון.

במעבד זה, בנוסף ללולאות for, כל התניה (פקודות if בשפת C) ממומשת ע"י פקודת קפיצה מותנית אחת. כניסה לתנאי ה- if (True) נחשבת ה- Taken מבחינת פקודת הקפיצה המותנית.

נתונה התוכנית הבאה, כאשר X הוא מערך int באורך 1000:

```
Count_div3 = 0 ;
for ( i = 0; i < 1000; i++) {
        if (X[i] % 3 == 0) count_div3++ ;
}
```

א. (5 נקודות) הסבירו במילים מה עושה התוכנית.

התוכנית עוברת על מערך בעל 1000 איברים, ומונה כמה מאיברי המערך מתחלקים ב- 3 ללא שארית.

ב. (6 נקודות) חשבו את כמות הזיכרון הנדרשת לחזאי. יש להתעלם בחישוב ממנגנון מדיניות ההחלפה.

לחזאי יש 4 כניסות, בכל כניסה יש לשמור את ה- tag (ה- pc), את כתובת היעד לקפיצה, את ההיסטוריה הלוקאלית ואת סט מכונות המצבים, ולכן:

$$|BTB| = 4 \cdot [32 + 32 + 2 + 2 \cdot 2^2] = 296[bits]$$

התקבלה גם ההנחה כי בכל כניסה של ה- BTB ישנו מספר שלם של בתים, ולכן:

$$|BTB| = 4 \cdot \left\lceil \frac{[32 + 32 + 2 + 2 \cdot 2^2]}{8} \right\rceil = 40[Bytes]$$

ג. (10 נקודות) השלימו את הטבלה הבאה לגבי החיזוי של הוראת ה- if עבור 6 האיטרציות הראשונות, כאשר: $X = (0, 1, 2, 3, \dots, 999)$

מאחר ובשאלה לא נתון קוד האסמבלי של התוכנית, נאמר לנבחנים במהלך הבחינה כי ניתן להניח (עבור כל סעיפי השאלה) שבין כל שתי פקודות של קפיצה מותנית חולפים מספיק

מחזורי שעון כך שה- BTB מספיק להתעדכן. הנחה זו נועדה כדי להקל על הנבחנים וכדי ליצור אחידות בפתרון הבחינה.

i	X[i]	Taken/ Not Taken	ערך ההיסטוריה לפני החיזוי	מצב המכונה לפני החיזוי	חיזוי בפועל	נכונות החיזוי
0	0	T	00	01=WNT	NT	Miss
1	1	NT	01	01=WNT	NT	Hit
2	2	NT	10	01=WNT	NT	Hit
3	3	T	00	10=WT	T	Hit
4	4	NT	01	00=SNT	NT	Hit
5	5	NT	10	00=SNT	NT	Hit

ד. (10 נקודות) **בסעיף זה בלבד**, משנים את סדר האיברים במערך X לסדר אקראי. האם מספר החיזויים השגויים יגדל/יקטן/לא ישתנה? נמקו תשובתכם!

היות והצלחת החיזוי מבוססת על תבנית קבועה בעלת מחזוריות של 3, לכן חזאי בעל 2 סיביות היסטוריה יכול להגיע לחיזוי מושלם (לאחר שלב האימון של מכונות המצבים). אולם, שינוי סדר האיברים של המערך באופן אקראי יפגע במבנה המחזורי, וממילא מספר החיזויים השגויים יגדל (הערה: אמנם, קיימת אפשרות שיצא לנו מבנה עם מחזוריות של 3, אך זה קורה בהסתברות קטנה מאוד, ולכן, רוב הסיכויים שמספר החיזויים השגויים יגדל, ולכל היותר הוא לא יקטן).

כעת, משנים את התוכנית באופן הבא:

```
count_div6 = 0 ;
count_div2 = 0 ;
count_div3 = 0 ;
for ( i = 0; i < 1000; i++) {
        if (X[i] % 6 == 0) count_div6++ ;
        if (X[i] % 2 == 0) count_div2++ ;
        if (X[i] % 3 == 0) count_div3++ ;
}
```

ה. (10 נקודות) מריצים את התוכנית אין-סוף פעמים. חשבו את ה- misprediction rate של החזאי במצב היציב (ניתן להזניח את ההחטאות של האיטרציות הראשונות), כאשר המערך X הוא כמו בסעיף ג'.

עבור התנאי של חלוקה ב- 6: ישנה תבנית בעלת מחזוריות של 6: T, NT, NT, NT, NT, NT. מכונות המצבים המתאימות לוקטור ההיסטוריה 01 ו- 10 יגיעו לחיזוי מושלם, אולם, מכונת המצבים המתאימה להיסטוריה של 00 תבצע חיזוי שגוי כאשר היא אמורה לחזות T, משום שהיא תמיד תחזה NT. שימו-לב, כי במהלך מעבר על הווקטור X כולו החזאי יטעה 167 פעמים מתוך 666 גישות למכונת המצבים המתאימה להיסטוריה 00. לכל אחת משתי

המכונות האחרות (בעלות החיזוי המושלם) ניגשים 167 פעמים. לכן, בסה"כ עבור התנאי הבודק חלוקה ב- 6:

$$misprediction\ rate_{\%6} = \frac{167+0+0}{1000} = \frac{167}{1000} = 0.167$$

עבור התנאי של חלוקה ב- 2: ישנה תבנית בעלת מחזוריות של 2: T, NT, T, NT. היות וישנו וקטור היסטוריה לוקאלי באורך 2, החזאי ייתן חיזוי מושלם במצב היציב. ולכן:

$$misprediction\ rate_{\%2} = 0$$

עבור התנאי של חלוקה ב- 3: ישנה תבנית בעלת מחזוריות של 3: T, NT, NT, T, NT, NT. אולם, בסוף איטרציה של תוכנית תהיה שבירה של הרצף (משום שגם המספר 999 מתחלק ב- 3 ללא שארית, וגם המספר 0 מתחלק ב- 3 ללא שארית). לכן, מכונת המצבים המתאימה לווקטור ההיסטוריה 11 תהיה בשימוש פעם אחת בכל איטרציה של התוכנית, והיא תמיד תחזה נכון. מכונת המצבים המתאימה לווקטור ההיסטוריה 00 תתייצב על ST ותחזה נכון תמיד. מכונת המצבים המתאימה לווקטור ההיסטוריה 01 תתייצב על SNT ותחזה נכון תמיד. למכונה 10 ניגשים 333 פעמים במהלך כל איטרציה של התוכנית, מתוכן החזאי יטעה פעם אחת, ולכן:

$$misprediction\ rate_{\%3} = \frac{0+0+1+0}{1000} = \frac{1}{1000} = 0.001$$

עבור תנאי הקפיצה של הלולאה: ישנה תבנית בעלת מחזוריות של 1000, כאשר 999 פעמים התנאי יילקח (T) ופעם אחת התנאי לא יילקח (NT). לכן, החזאי יטעה פעם אחת מתוך 1000 גישות אליו, ולכן:

$$misprediction\ rate_{for} = \frac{1}{1000}$$

בסה"כ נקבל:

$$Total\ misprediction\ rate = \frac{\frac{167}{1000}+0+\frac{1}{1000}+\frac{1}{1000}}{4} = \frac{169}{4000} = 0.04225$$

כעת, מחליפים את החזאי לחזאי עם היסטוריה גלובלית באורך 2 סיביות. מכונות המצבים נותרו ללא שינוי (פרטיות).

ו. (10 נקודות) מה תהיה ההשפעה על מספר החיזויים השגויים במצב היציב לכל אחת מפקודות ה- if ביחס לסעיף הקודם (יעלה/ירד/לא ישתנה)? הסבירו תשובתם.

לצורך הבנת הפתרון של סעיף זה, ניתן להיעזר בטבלה בסוף הקובץ.

עבור התנאי של חלוקה ב- 6: ניתן לראות כי הפקודה תשתמש במכונות המצבים המתאימות לווקטורי ההיסטוריה: 10, 11 ו- 01, כאשר מכונת המצבים המתאימה לווקטור ההיסטוריה 10 תהיה בשימוש פעם אחת בלבד בכל איטרציה של התוכנית, והיא תמיד תחזה נכון. מכונת המצבים המתאימה לווקטור ההיסטוריה 11 תתייצב על SNT ותחזה נכון תמיד, ואילו מכונה 01 תיתן חיזוי שגוי כאשר היא אמורה לחזות Taken ובפועל תחזה Not-Taken. למכונה 10 ניגשים רק פעם 1 מתוכך כל ה- 1000 פעמים שמבצעים את תנאי החלוקה ב- 6, למכונה 11 ניגשים 333 פעמים, ולמכונה 01 ניגשים 666 פעמים, ומתוכם מחטיאים 166 פעמים. לכן, ה- misprediction rate יקטן במעט ביחס לסעיף הקודם:

$$misprediction\ rate_{\%6} = \frac{0+0+166}{1000} = 0.166$$

עבור התנאי של חלוקה ב- 2: ניתן לראות כי הפקודה תשתמש במכונות המצבים: 01, 11 ו- 10, כאשר מכונת המצבים המתאימה לווקטור ההיסטוריה 01 תהיה בשימוש פעם אחת בלבד בכל איטרציה של התוכנית, והיא תמיד תחזה נכון. מכונה 11 תתייצב על ST ותחזה נכון תמיד, אולם, מכונה 10 אמורה לחזות רצפים של NT, T, NT,T,NT ולכן ה- misprediction rate שלה יגדל ביחס לסעיף הקודם.

עבור התנאי של חלוקה ב-3: ניתן לראות כי הפקודה תשתמש במכונות המצבים 11, 01 ו- 00, כאשר מכונה 11 תתייצב על ST ותחזה נכון תמיד, מכונה 01 תתייצב על SNT ותחזה נכון תמיד, אולם, מכונה 00 אמורה לחזות רצפים של NT, T, NT ולכן ה- misprediction rate שלה יגדל ביחס לסעיף הקודם.

עבור תנאי הקפיצה של הלולאה (לא נדרש בשאלה): ניתן לראות בטבלת העזרת בסוף הקובץ כי התנאי של ה- for ישתמש בכל מכונות המצבים שלו, ובמצב היציב כולן תהינה במצב ST, ולכן יהיה לו חיזוי מושלם, למעט חיזוי אחד שגוי פעם באלף גישות אליו. לכן, ה- misprediction rate שלו יהיה כמו בסעיף הקודם.

ז. (9 נקודות) עלתה הצעה לשנות את סדר ביצוע התנאים בתוכנית על מנת לשפר את ביצועי החיזוי של החזאי הראשון (תנאי החלוקה ב- 6), כאשר החזאי הוא כמו בסעיף הקודם. האם שיפור כזה אפשרי? אם כן, ציינו מהו השינוי ומהו ה- misprediction rate של החזאי לאחר השיפור. אם לא, נמקו.

הדבר אפשרי: אם נבדוק קודם את תנאי החלוקה ב- 2 וב- 3, ורק אח"כ נבדוק את תנאי החלוקה ב- 6, הרי שהחזאי ילמד לחזות חיזוי מושלם, שהרי אם המספר מתחלק גם ב- 2 וגם ב- 3 ללא שארית (במקרה כזה, וקטור ההיסטוריה יהיה 11 כאשר נגיע לתנאי החלוקה ב- 6) אזי המספר מתחלק גם ב- 6 ללא שארית, ומכונת המצבים מתאימה תתייצב על ST. בכל מקרה אחר (וקטור ההיסטוריה יהיה שונה מ- 11) המספר לא מתחלק ב- 6 ללא שארית, ומכונות המצבים המתאימות תתייצבנה על SNT.

ניתוח מפורט עבור שאלה 1 סעיף ד' (לא נדרש בבחינה):

#	IF	ID	RS	Exe
1	1,2			
2	3,4	1,2		
3	5,6	3,4	1,2	
4	11,12	5,6	3,4	1,2
5	13,14	11,12	3,5,6	1,2,4
6	15,16	13,14	5,11,12	3,6
7	21,22	15,16	12,13,14	5,11
8	23,24	21,22	13,15,16	11,12,14
9	25,26	23,24	13,15,21,22	12,16
10	31,32	25,26	15,22,23,24	13,21
11	33,34	31,32	23,24,25,26	21,15,22
12	35,36	33,34	23,25,26,31,32	22,24
13	41,42	35,36	25,31,32,33,34	23,26
14	43,44	41,42	32,33,34,35,36	25,31
15	45,46	43,44	33,35,36,41,42	31,32,34
16	51,52	45,46	33,35,42,43,44	32,36,41
17	53,54	51,52	35,43,44,45,46	41,33,42
18	55,56	53,54	43,45,46,51,52	42,35,44
19	61,62	55,56	45,51,52,53,54	43,46
20	…	61,62	52,53,54,55,56	45,51

ניתוח מפורט עבור שאלה 1 סעיף ה' (לא נדרש בבחינה):

#	IF	ID	RS	Exe
1	1,2			
2	3,4	1,2		
3	5,6	3,4	1,2	
4	11,12	5,6	3,4	1,2
5	13,14	11,12	3,5,6	1,2,4
6	15,16	13,14	3,5,11,12	1,2,6
7	21,22	15,16	3,5,13,14	1,2,11,12
8	23,24	21,22	3,5,13,15,16	1,2,11,12,14
9	25,26	23,24	3,5,13,15,21,22	1,2,11,12,16
10	31,32	25,26	3,5,13,15,23,24	1,2,11,12,21,22
11	33,34	31,32	3,5,13,15,23,25,26	1,2,11,12,21,22,24
12	35,36	33,34	3,5,13,15,23,25,31,32	1,2,11,12,21,22,26
13	41,42	35,36	3,5,13,15,23,25,33,34	1,2,11,12,21,22,31,32
14	43,44	41,42	3,5,13,15,23,25,33,35,36	1,2,11,12,21,22,31,32,34
15	45,46	43,44	3,5,13,15,23,25,33,35,41,42	1,2,11,12,21,22,31,32,36
16	51,52	45,46	3,5,13,15,23,25,33,35,43,44	1,2,11,12,21,22,31,32,41,42
17	53,54	51,52	3,5,13,15,23,25,33,35,43,45,46	1,2,11,12,21,22,31,32,41,42,44
18	55,56	53,54	3,5,13,15,23,25,33,35,43,45,51,52	1,2,11,12,21,22,31,32,41,42,46
19	61,62	55,56	3,5,13,15,23,25,33,35,43,45,53,54	1,2,11,12,21,22,31,32,41,42,51,52
20	63,64	61,62	3,5,13,15,23,25,33,35,43,45,53,55,56	1,2,11,12,21,22,31,32,41,42,51,52,54
21	65,66	63,64	3,5,13,15, … ,53,55,61,62	1,2,11,12, … ,51,52,56
22	71,72	65,66	3,5,13,15, … ,53,55,63,64	1,2,11,12, … ,51,52,61,62
23	73,74	71,72	3,5,13,15, … ,53,55,63,65,66	1,2,11,12, … ,51,52,61,62,64
24	75,76	73,74	3,5,13,15, … ,53,55,63,65,71,72	1,2,11,12, … ,51,52,61,62,66
25	81,82	75,76	3,5,13,15, … ,53,55,63,65,73,74	1,2,11,12, … ,51,52,61,62,71,72
26	83,84	81,82	5,13,15, … ,53,55,63,65,73,75,76	11,12,21,22 … ,51,52,61,62,71,72,3,74
27	85,86	83,84	13,15, … ,53,55,63,65,73,75,81,82	11,12,21,22 … ,51,52,61,62,71,72,5,76
28	91,92	85,86	13,15, … ,73,75,83,84	11,12,21,22 … ,51,52,61,62,71,72,81,82
29	93,94	91,92	15,23 … ,73,75,83,85,86	21,22,31,32, …,71,72,81,82,13,84
30	95,96	93,94	23,25, … ,73,75,83,85,91,92	21,22,31,32, …,71,72,81,82,15,86
31	…	95,96	23,25, … ,73,75,83,85,93,94	21,22,31,32, …,71,72,81,82,91,92

טבלת עזרת לשאלה 2 סעיף ו':

Branch	X[i]	T/NT	History	remarks	
mod6	999	0		$999\ mod\ 6\ \neq\ 0$	
mod2	999	0		$999\ mod\ 2\ \neq\ 0$	
mod3	999	1	0 0	$999\ mod\ 3\ =\ 0$	
for()	999	0	0 1	Miss of the last iteration	
mod6	0	1	1 0	New iteration of the program	Hit
mod2	0	1	0 1		Hit
mod3	0	1	1 1		
for()	0	1	1 1		
mod6	1	0	1 1		
mod2	1	0	1 0		
mod3	1	0	0 0		
for()	1	1	0 0		
mod6	2	0	0 1		
mod2	2	1	1 0		
mod3	2	0	0 1		
for()	2	1	1 0		
mod6	3	0	0 1		
mod2	3	0	1 0		
mod3	3	1	0 0		
for()	3	1	0 1		
mod6	4	0	1 1		
mod2	4	1	1 0		
mod3	4	0	0 1		
for()	4	1	1 0		
mod6	5	0	0 1		
mod2	5	0	1 0		
mod3	5	0	0 0		
for()	5	1	0 0		
mod6	6	1	0 1		Miss
mod2	6	1	1 1		
mod3	6	1	1 1		
for()	6	1	1 1		
mod6	7	0	1 1		
mod2	7	0	1 0		
mod3	7	0	0 0		
for()	7	1	0 0		
mod6	8	0	0 1		
mod2	8	1	1 0		
mod3	8	0	0 1		
for()	8	1	1 0		
mod6	9	0	0 1		
mod2	9	0	1 0		

mod3	9	1	0 0		
for()	9	1	0 1		
mod6	10	0	1 1		
mod2	10	1	1 0		
mod3	10	0	0 1		
for()	10	1	1 0		
mod6	11	0	0 1		
mod2	11	0	1 0		
mod3	11	0	0 0		
for()	11	1	0 0		
mod6	12	1	0 1		Miss
mod2	12	1	1 1		
mod3	12	1	1 1		
for()	12	1	1 1		