|  |  |  |
| --- | --- | --- |
| **BAR-ILAN UNIVERSITY**  Engineering Faculty |  | אוניברסיטת בר-אילן הפקולטה להנדסה |

**מבנה מחשבים ספרתיים 83-301**

# תשע"ה סמס' ב' מועד א'

|  |  |  |  |
| --- | --- | --- | --- |
| שם הקורס | מבנה מחשבים ספרתיים | | |
| מספר הקורס | 83-301 | | |
| שם המרצה | פרופ' שמואל וימר + מר עמית מנדלבאום | | |
|  | תשע"ה | סמסטר ב' | מועד א' |
| משך הבחינה | שלש שעות | | |
| חומר עזר | שני ספר הקורס של המחברים Patterson & Hennessy + שקפי הקורס והתרגול בלבד (כולל הערות על גביהם).  כל חומר אחר, כולל תרגילים, אמצעים אלקטרונים, מחשבונים, וכו', אסורים בשימוש.  **יש לצרף את שאלוני הבחינה למחברת**. | | |
|  | יש לענות על כל השאלות. כל תשובה יש לנמק ולהסביר. כל תשובה מספרית מחייבת את הצגת דרך החישוב.  סה"כ הנקודות האפשריות 110. ציון הבחינה לא יעלה על 100.  **יש לכתוב בעט בלבד. כתיבה בעפרון לא תיבדק**. | | |

**בהצלחה!**

**שאלה א' (25 נק')**

1. (2 נק') ידוע כי הוספת Cache כמתווך בין ה- CPU לזיכרון הראשי מגדילה את זמן קבלת המידע ישירות מהזיכרון הראשי, מדוע בכל זאת הוספת Cache מקטינה את זמן הגישה לזיכרון?

**תשובה**: אמנם זמן קבלת המידע מהזיכרון עולה אך לאחר מכן, בזכות השימוש בעיקרון הלקאליות, מידע שהיה באותו הבלוק שהובא מהזיכרון ל – Cache נקרא שוב ללא גישות נוספות לזיכרון הראשי ולכן בסך הכל זמן הגישה לזיכרון מתקצר.

**טעות נפוצה:** לא לאזכר את עקרון הלוקאליות

1. (5 נק') נתונים שני מעבדים הפועלים באותו תדר ומריצים אותה תכנית בדיוק.

במעבד הראשון זמן הגישה לזכרון הראשי במקרה של CACHE MISS הינו 30 ננושניות, ואילו CACHE HIT אורך 3 ננושניות. ידוע ש CACHE HIT קורה ב 80% מהזמן.

במעבד השני זמן הגישה לזכרון הראשי במקרה של CACHE MISS הינו 50 ננושניות, ואילו CACHE HIT אורך 2 ננושניות. ידוע ש CACHE HIT קורה ב 90% מהזמן.

חשב בצורה מלאה את ביצועי הזכרון של שני המעבדים. לאיזה מהם ביצועים טובים יותר?

**Answer:**

In the 1st processor 80% of the time the access will be 3 ns while the rest of the time (20%) the access time will be 30 ns. The average is

(0.8 × 3 ns) + (0.2 × 30 ns) = 2.4 ns + 6 ns = 8.4 ns

In the 1st processor 90% of the time the access will be 2 ns while the rest of the time (10%) the access time will be 50 ns. The average is

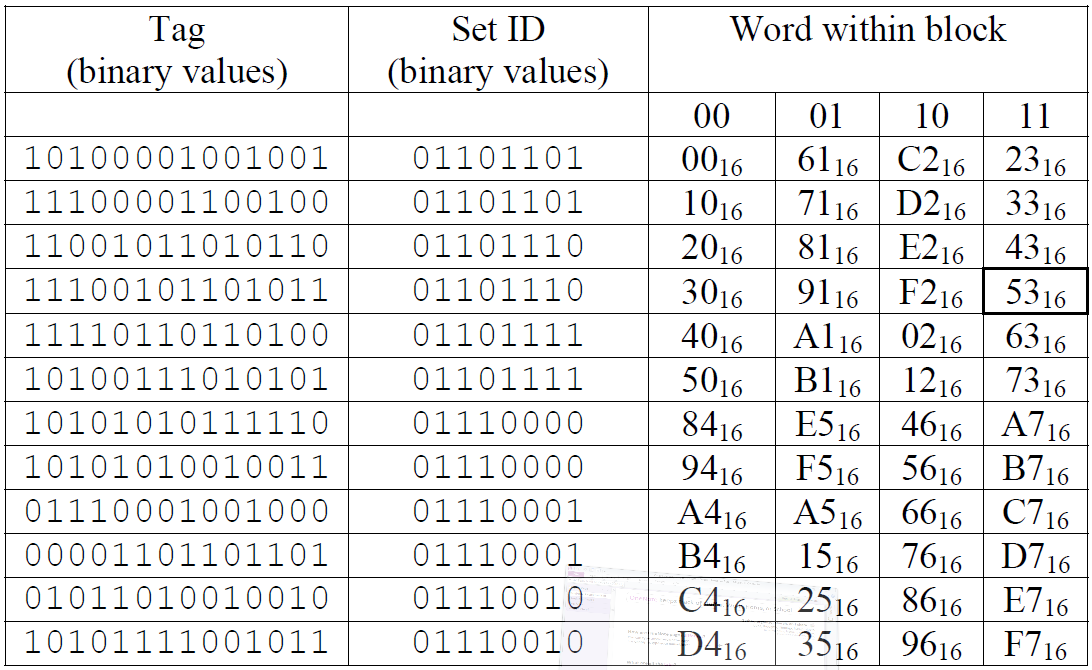
(0.9 × 2 ns) + (0.1 × 50 ns) = 1.8 ns + 5 ns = 6.8 ns

**טעות נפוצה:** חישוב Hit-time ב- 100% מהמקרים

1. (3 נק') איזה מבין שלשת סוגי זכרון המטמון (**DIRECT**, FULLY ASSOCIATIVE, N-WAY) איננו נדרש ל ?REPLACEMENT ALGORITHM **מדוע?**

**תשובה**: ב-Cache מסוג Direct כל כתובת ממופה לתא אחד בלבד ב-cache ולכן כאשר יש צורך לחליף מידע בתא מסוים אין "בחירה" איזה תא להחליף ולכן לא נדרש אלגוריתם החלפה.

1. (15 נק') לפניך חלק מתוך CACHE של מעבד כלשהו. בכל אחד מהסעיפים שלהלן צרף לתשובתך **הסבר מלא וחישוב מדוייק**.



1. מאיזה סוג הCACHE הנ"ל? (DIRECT, FULLY ASSOCIATIVE, N-WAY)

**Answer**: It is a 2-way set associative cache, i.e., there are two lines per set.

**טעות נפוצה**: 4-way כי יש 4 בתים בבלוק

1. מה גודל מרחב הכתובות של המעבד?

**תשובה**: ישנן 24 ביטים בכתובת סה"כ (14 + 8 +2) לכןBytes

1. כמה בלוקים יש במרחב הכתובות?

**תשובה**: בכל בלוק ישנם 4 בתים, לכן Blocks

1. מה גדלו של ה CACHE בבתים? (BYTES)

**Answer**: Since it is a 2-way set associative cache, and since there is an 8-bit tag, there are lines. Each line has 4 bytes, so the cache has a total of 2048 bytes.

**טעות נפוצה**: חישוב לפי הגודל שמופיע בטופס הבחינה (למרות שכתוב במפורש שזה רק חלק).

1. נדרש לקרוא ממרחב הזיכרון את הבית BYTE)) הנמצא בכתובת 0x7121C5. האם ניתן לדעת מה ערכו של הנתון בכתובת הנ"ל? ובמידה וכן, מהו ערכו?

**Answer**:

First, convert 7121C5 to binary. 7121C516 = 111000100100001110001012

The last two bits, 01, identify the word position within the block. 01 means that it is in the second column of data.

The next eight bits, 01110001, should identify the set. It identifies the set consisting of the third and fourth rows from the bottom.

The fourth row from the bottom has the matching tag of 0111000100100. Therefore, the data is in the second column of the fourth row from the bottom, i.e., **A5**.

**שאלה ב' (50 נק')**

מהם מצבי טבלאות ה RS, ROB, וה FP RF (FLOATING POINT REGISTER FILE) המצורפות עבור מעבד ספקולטיבי המממש את אלגוריתם TOMASULO, אשר מריץ את התכניות שלהלן, תחת ההנחות הבאות:

* רק פקודה אחת יכולה לצאת (ISSUE) בכל מחזור שעון.
* ל ROB ישנן 8 כניסות (SLOTS) והוא משמש גם בתור חוצץ (BUFFER) לLOAD ול .STORE
* כל היחידות הפונקציונליות (FU) הינן FULLY PIPELINED.
* ישנן 2 יחידות RS לכפל FP MULTIPLY)), 3 יחידות לחבור FP ADD)) ו 3 יחידות RS לשלמים, המשמשות גם ל LOAD ו STORE.
* חריגות (EXCEPTIONS) אינן קורות.
* השיהוי בבצוע (EXECUTION) פעולות בשלמים הינו מחזור שעון אחד.
* להוציא STRUCTURAL HAZARD, פעולת LOAD יוצאת לדרך במחזור שעון אחד, מבוצעת בזה שאחריו, וכותבת בשלישי. פעולה התלויה ב LOAD יכולה להתבצע (EXECUTE) ברביעי.
* השיהוי בבצוע (EXECUTION) חבור FP הינו 2 מחזורי שעון וכפל FP 4 מחזורי שעון.
* במקרה של קונפליקט בכתיבה ל CDB, הקדימות (PRIORITY) לפקודה שיצאה לדרך מוקדם יותר.
* בצוע (EXECUTION) של פקודות תלויות יכול להתחיל במחזור השעון לאחר שנתוניהן משודרים ב CDB.
* מניחים שטבלאות ה RS, ROB ו FP RF הינן ריקות ופנויות בעת תחילת בצוע התכניות שלהלן.
* העמודה "VALUE" מתעדכנת בעת שידור הערך ב CDB.

**דרישה חשובה בתשובות לשאלה:** בכל שינו ממצב "BUSY" ל ""NOT BUSY, נדרש כמובן לעדכן את העמודה המתאימה, אבל **לא למחוק** את תוכן שאר העמודות (אלא אם כן הדבר נדרש כתוצאה מפקודה אחרת התופסת מקום זה).

**טעויות כלליות נפוצות:**

א. רישום שדה Value גם עבור פקודות שלא עשו Write.

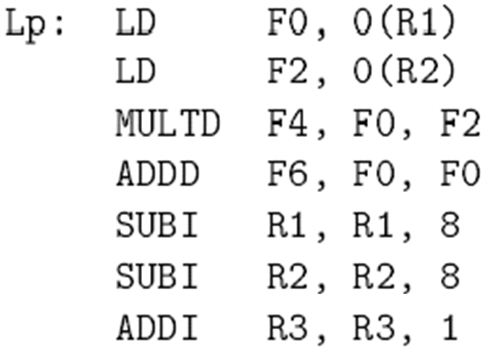
ב. רישום שדות Q עבור פקודות שעשו (או במהלך) Execute

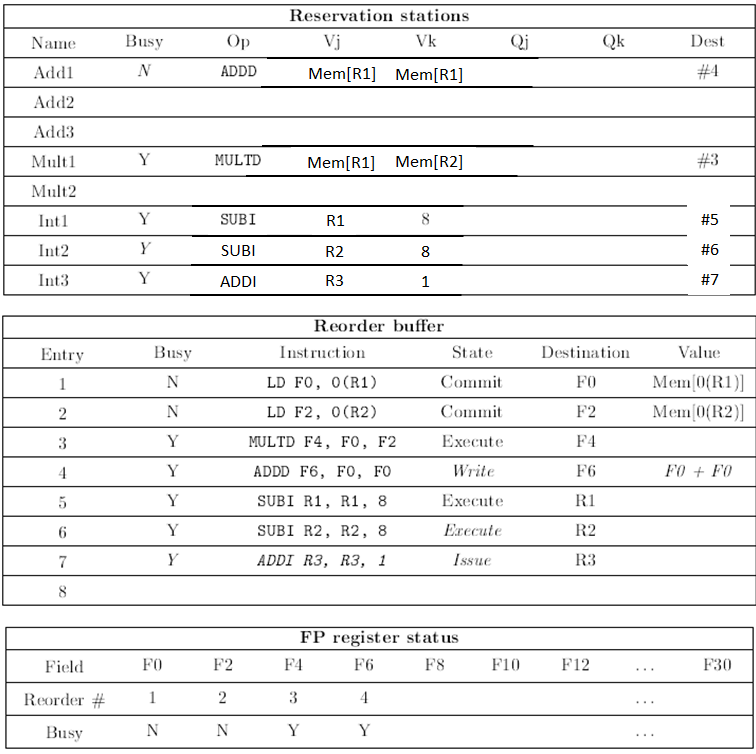
ג. מחיקת רגיסטרים מה-Register status כאשר הם לא Busy

ד.חוסר עדכון של שדות ה- Vj ו-Vk כאשר הערך שלהם אמור להתעדכן.

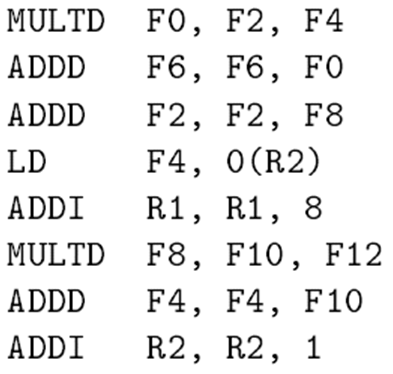
סעיף ב' טעות נפוצה מאוד – פקודה לפני אחרונה ב- Execute במקום ב- Issue

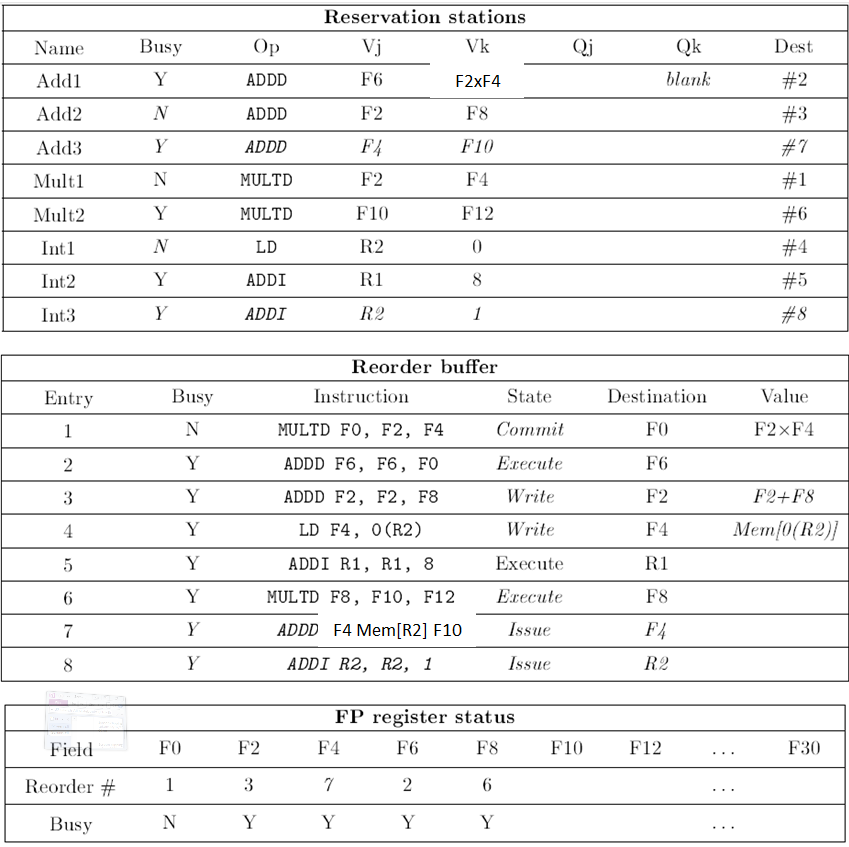
1. (25 נקודות) רשום באופן מלא את הטבלאות המצורפות בעת השגור של הפקודה האחרונה בקוד שלהלן





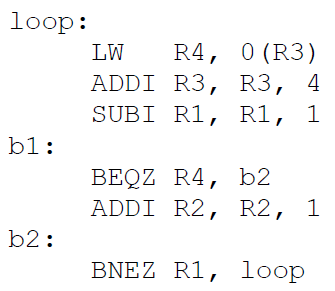
1. (25 נקודות) רשום באופן מלא את הטבלאות המצורפות בעת השגור של הפקודה האחרונה בקוד שלהלן





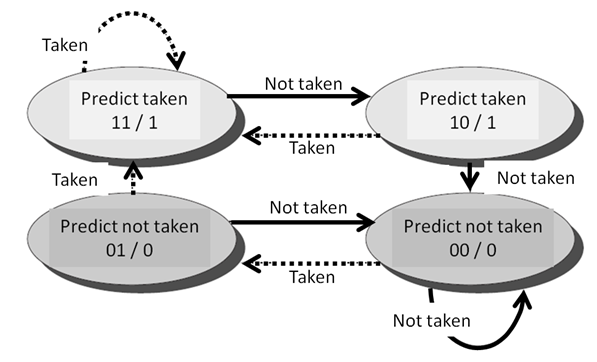
**שאלה ג' (35 נקודות)**

נתונה התכנית שלהלן



נתונים ערכי ההתחלה הבאים: , ,  *(מצביע לראש מערך של מספרים שלמים).*

*ברשותנו מעבד* MIPS *בעל מנגנון חיזוי קפיצות המתואר להלן. הניחו ש* ו אינם מתנגשים בטבלת היסטורית הקפיצות.



1. (5 נקודות) מה בדיוק מבצעת התכנית הנ"ל? מהו ערכו של בעת היציאה מהלולאה?

**תשובה**: התוכנית מקבלת מערך P וסופרת כמה מבין n האיברים הראשונים בו שונים מ-0. בסיום התוכנית R2 מכיל את כמות האיברים ששונים מ-0

**טעות נפוצה**: לא מתייחסים ל –n

1. (10 נקודות) נניח שהקלט לתכנית הינו ו . **מלא את טבלה א' המצורפת במלואה**. העמודה PC שבטבלה מציינת את הפקודה בה נמצא ה PC. העמודות ו מציינות את מצב מכונות המצבים המתאימות, N מציין NOT TAKEN, ואילו T מציין TAKEN. מצב מכונת המצבים בעת קבלת הניחוש מצויין במודגש, ואילו המצב הבא מצויין באלכסון. **מהו מספר חיזויי הקפיצה השגויים**?

**תשובה**: (ראה טבלה ליד הטבלה המקורית) מספר החיזויים השגויים : 7

**טעות נפוצה (גם בסעיף ג')** לא שמים לב ש-R1 יורד ב-1 **לפני** הקפיצות ולכן ממלאים את 2 השורות האחרונות בטבלה (למרות שאמורות להישאר ריקות)

1. (10 נקודות) מוסיפים כעת לטבלת חיזוי הקפיצות סיבית של היסטוריה כאשר הסיבית 0 פירושו שלא קרתה קפיצה ו 1 כאשר קרתה קפיצה. **מלא את טבלה ב' המצורפת במלואה**. **מהו מספר חיזויי הקפיצה השגויים**?

**תשובה**: (ראה טבלה ליד הטבלה המקורית) מספר חיזויים שגויים: 5

**טעות נפוצה:** חוסר הבנה של משמעות של ביט ההיסטוריה

1. (10 נקודות) השווה בין חיזוי הקפיצות בסעיף ב' ובסעיף' ג' לעיל.

* באיזה שלב של התכנית (התחלה, אמצע, סוף) קורות מירב השגיאות בחיזוי הקפיצה, ומדוע?

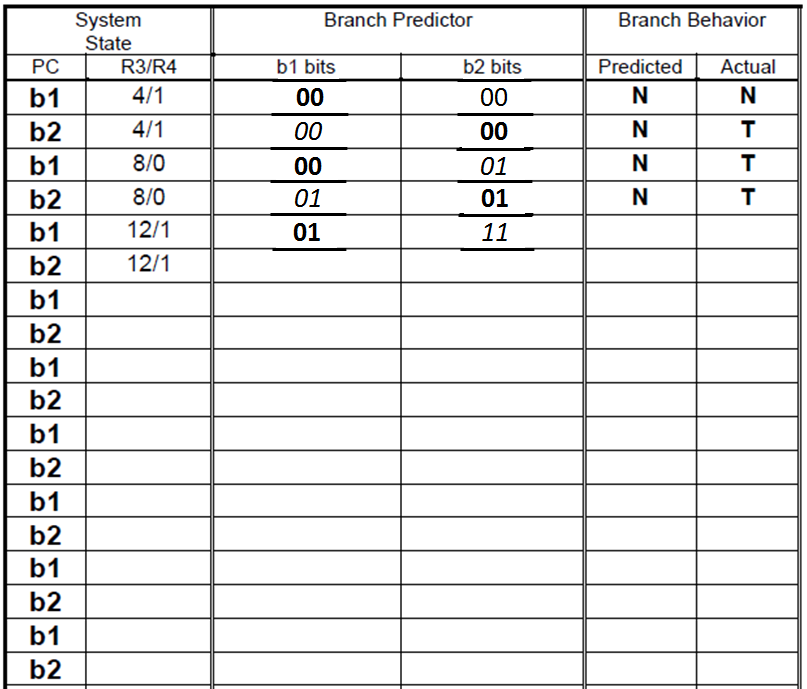
**תשובה**: בהתחלה. עבור הלולאה השנייה (שתמיד נלקחת עד הסוף) , לוקח זמן לחוזה להתייצב ולכן יש שגיאות בהתחלה. עבור הלולאה הראשונה (במקרה שיש ביט של היסטוריה) כנ"ל

* האם הוספת סיבית נוספת של היסטורית קפיצות תשנה את המצב עבור תכנית זאת והנתונים הללו (תשובה ללא הסבר ונימוק לא תתקבל)?

**תשובה**: לא, הלולאה ראשונה נלקחת ולא נלקחת לסירוגין. לכן מספיק לנו רק ביט אחד של היסטוריה, ביט נוסף לא יעזור כיוון שתמיד ניפול בסטים 01 ו 10 וממילא חזרנו למצב שבו יש לנו רק שני סטים (=ביט אחד של היסטוריה). עבור הלולאה השנייה ממילא אין שיפור במעבר לשני סטים ולכן ברור שהוספת ביט נוסף לא תשפר את המצב.

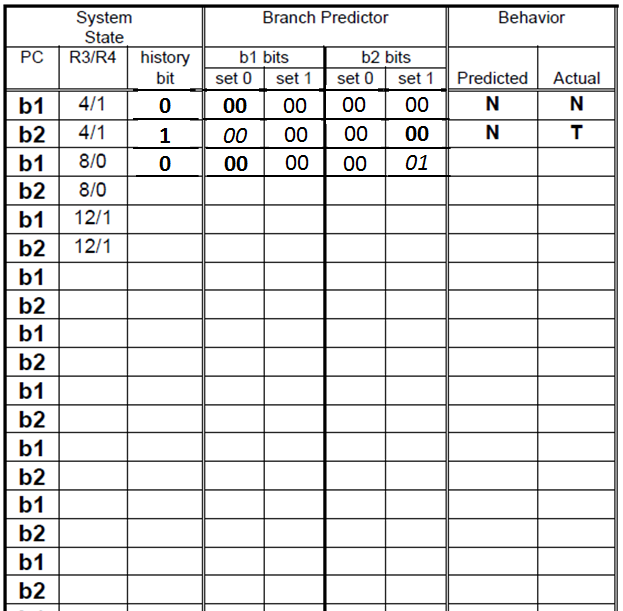
**טעות נפוצה:** חוסר התייחסות לנתונים (כלומר לעובדה שהקפיצה נלקחת ולא נלקחת לסירוגין בלולאה הראשונה)

**טבלה א'**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Branch behavior | | Predictor | | System state | |
| Actual | Predicted | B2 bits | B1 bits | R3/R4 | PC |
| N | N | 00 | 00 | 4/1 | B1 |
| T | N | 00 | 00 | 4/1 | B2 |
| T | N | 01 | 00 | 8/0 | B1 |
| T | N | 01 | 01 | 8/0 | B2 |
| N | N | 11 | 01 | 12/1 | B1 |
| T | T | 11 | 00 | 12/1 | B2 |
| T | N | 11 | 00 | 16/0 | B1 |
| T | T | 11 | 01 | 16/0 | B2 |
| N | N | 11 | 01 | 20/1 | B1 |
| T | T | 11 | 00 | 20/1 | B2 |
| T | N | 11 | 00 | 24/0 | B1 |
| T | T | 11 | 01 | 24/0 | B2 |
| N | N | 11 | 01 | 28/1 | B1 |
| T | T | 11 | 00 | 28/1 | B2 |
| T | N | 11 | 00 | 32/0 | B1 |
| N | T | 11 | 01 | 32/0 | B2 |

**טבלה ב'**



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Branch behavior | | Predictor | | | | History bit | System state | |
| Actual | Predicted | B2 bits | | B1 bits | |  | R3/R4 | PC |
| N | N | 00 | 00 | 00 | 00 | 0 | 4/1 | B1 |
| T | N | 00 | 00 | 00 | 00 | 1 | 4/1 | B2 |
| T | N | 01 | 00 | 00 | 00 | 0 | 8/0 | B1 |
| T | N | 01 | 00 | 00 | 01 | 1 | 8/0 | B2 |
| N | N | 11 | 00 | 00 | 01 | 1 | 12/1 | B1 |
| T | T | 11 | 00 | 00 | 01 | 1 | 12/1 | B2 |
| T | N | 11 | 00 | 00 | 01 | 0 | 16/0 | B1 |
| T | T | 11 | 00 | 00 | 11 | 1 | 16/0 | B2 |
| N | N | 11 | 00 | 00 | 11 | 1 | 20/1 | B1 |
| T | T | 11 | 00 | 00 | 11 | 1 | 20/1 | B2 |
| T | T | 11 | 00 | 00 | 11 | 0 | 24/0 | B1 |
| T | T | 11 | 00 | 00 | 11 | 1 | 24/0 | B2 |
| N | N | 11 | 00 | 00 | 11 | 1 | 28/1 | B1 |
| T | T | 11 | 00 | 00 | 11 | 1 | 28/1 | B2 |
| T | T | 11 | 00 | 00 | 11 | 0 | 32/0 | B1 |
| N | T | 11 | 00 | 00 | 11 | 1 | 32/0 | B2 |