מערכות ספרתיות ומבנה המחשב (044252)

סמסטר אביב תש"פ

בחינה סופית – מועד ב

פתרון

20 באוקטובר 2020

טור 1

מספר סטודנט

משך המבחן: 3 שעות (180 דקות). **תכננו את זמנכם היטב.**

חומר עזר: אין להשתמש בכל חומר עזר בכתב, מודפס או אלקטרוני, פרט לדפי העזר שיחולקו במהלך הבחינה ולמחשבון.

הנחיות והוראות:

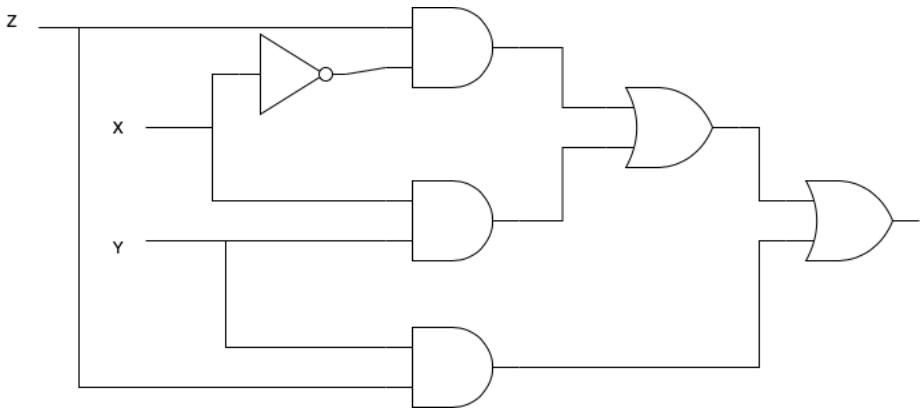
- הבחינה כתובה על גבי **18** עמודים כולל עמוד זה (**בדקו בתחילת הבחינה שלא חסרים לכם עמודים**).
- בתחילת הבחינה תקבלו חוברת בחינה, מחברת טיוטה, דפי עזר וטופס תשובות ממוחשב. בסיום הבחינה, החזירו את חוברת הבחינה וטופס התשובות הממוחשב בלבד.
- יש לענות על כל השאלות בגוף המבחן.
- אין **לתלוש או להפריד** דפים מחוברת הבחינה, ממחברות הטיוטה ומדפי העזר.
- יש לכתוב את התשובות באמצעות עט שחור או כחול בלבד. **אין לכתוב או לצייר בעט אדום**.
- רשמו את מספר הסטודנט שלכם על חוברת הבחינה (בראש עמוד זה), על דפי העזר, ועל כל מחברות הטיוטה. **ודאו כי על מחברת הבחינה ועל טופס התשובות האמריקאי מודבקת מדבקת הנבחן שלכם.**
- לא מורדות נקודות (אין "קנס") בגין תשובה שגויה. לכן, כדאי לסמן תשובה כלשהי לכל שאלה.
- ציון שאלות רב הברירה ייקבע על סמך סריקה ממוחשבת של טופס התשובות בלבד. **לא לשכוח לסמן בטופס התשובות הממוחשב את מספר הטור שלכם (מופיע בראש עמוד זה).**
- **את התשובות לשאלות הפתוחות יש לרשום בדף אשר מצורף בתחילת מחברת הבחינה**. לנוחיותכם, בכל שאלה פתוחה ישנו איזור לרישום הפתרון, אך תשובות אשר ירשמו באיזור זה לא יבדקו.
- אסור שימוש בכל חומר חיצוני מלבד מחשבון. אסורה העברת חומר כלשהו בין הנבחנים, ואסורה כל תקשורת עם אנשים אחרים או כל מקור מידע. האיסור חל על כל צורות התקשורת – מילולית, חזותית, כתובה, אלקטרונית, אלחוטית, או אחרת. בפרט, אין להחזיק בטלפון סלולארי.

בהצלחה!

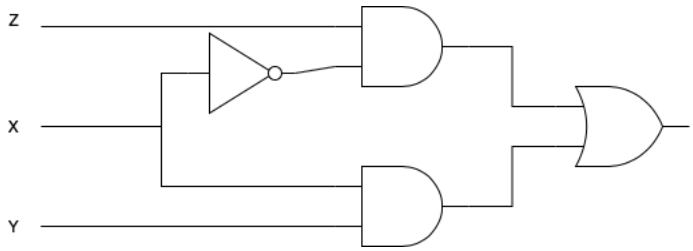
שאלה 1 (5 נקודות)

נתונה הפונקציה $f(x,y,z)$. הפונקציה מומשה על ידי המעגלים הצירופים הבאים:

מימוש 1:



מימוש 2:



זמני השהייה של כל השערים זהים ושונים מ-0.

בחרו את המשפט הנכון:

א. עבור שני המימושים יתכן כי יתרחש הבהוב (HAZARD) סטטי עם שינוי ערכו של המשתנה x (לשני הכיוונים), כאשר ערכם של y ו-z קבוע ויציב על 1 ($y = z = 1$).
ב. הבהוב (HAZARD) סטטי יתכן רק במימוש מספר 2, עם שינוי ערכו של המשתנה x (לשני הכיוונים), כאשר ערכם של y ו-z קבוע ויציב על 1 ($y = z = 1$).
ג. הבהוב (HAZARD) סטטי יתכן רק במימוש מספר 2, עם שינוי ערכו של המשתנה x מהערך 1 אל הערך 0, אך לא בכיוון ההפוך (מהערך 0 אל הערך 1), כאשר ערכם של y ו-z קבוע ויציב על 1 ($y = z = 1$).
ד. הבהוב (HAZARD) סטטי יתכן רק במימוש מספר 2, עם שינוי ערכו של המשתנה x מהערך 0 אל הערך 1, אך לא בכיוון ההפוך (מהערך 1 אל הערך 0), כאשר ערכם של y ו-z קבוע ויציב על 1 ($y = z = 1$).
ה. הבהוב (HAZARD) סטטי יתכן רק במימוש מספר 2, עם שינוי ערכו של המשתנה x (לשני הכיוונים), כאשר ל y ו-z ערכים לוגים כלשהם (קבועים ויציבים).

פתרון:

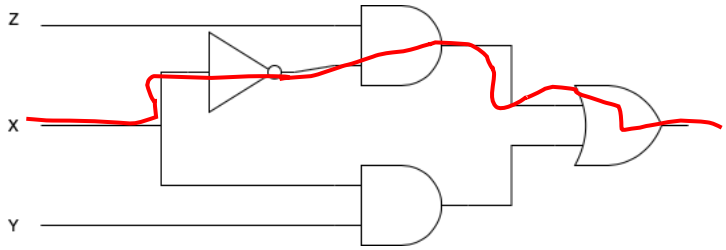
תשובה ג'.

שני המימושים מממשים את אותה הפונקציה אך בייצוגים אחרים כסכום של מכפלות:

מימוש 1: $x'z + xy + zy$

מימוש 2: $x'z + xy$

במימוש השני, מעבר של x מהערך 1 ל-0, מאלץ מעבר במסלול ארוך יותר (דרך המהפך) עד אשר השינוי מחלחל אל היציאה (המסלול האיטי מסומן באיור מטה). כיוון שהמסלול הארוך הוא זה שמספק את הערך אחד, ישנו שינוי זמני של המוצא אל הערך 0 בעקבות התאפסותו של המסלול הקצר. במימוש הראשון ישנה מכפלה אשר מכסה את המעבר בין שני ה-minterms אשר עשויים לגרום להבהוב כאשר $y = z = 1$ ו-x משנה את ערכו (בפועל זה גורם לכך שאין תלות ב-x במקרה זה).



שאלה 2 (5 נקודות)

נתונים זמני המחזור של כל ארכיטקטורות מעבדי RISC-V אשר נלמדו בקורס:

$T_{single\ cycle} = 6ns$

$T_{Multicycle} = 1.25ns$

$T_{pipeline} = 1ns$

מעבד ה-Pipelined RISC-V מכיל מנגנון forwarding מלא.

במידה ונריץ על כל אחת מן הארכיטקטורות תוכנית אשר מורכבת מפקודות R-type בלבד, מהו יחס ערכי ה-throughput של הארכיטקטורות השונות? הניחו כי מספר הפקודות גדול מאוד.

א. $troughput_{single\ cycle} > troughput_{Multicycle} > troughput_{pipeline}$
ב. $troughput_{pipeline} > troughput_{single\ cycle} > troughput_{Multicycle}$
ג. $troughput_{Multicycle} > troughput_{single\ cycle} > troughput_{pipeline}$
ד. $troughput_{pipeline} > troughput_{Multicycle} > troughput_{single\ cycle}$
ה. $troughput_{pipeline} = troughput_{Multicycle} > troughput_{single\ cycle}$

פתרון:

תשובה ד'.

נניח כי ישנן n פקודות בתוכנית.

עבור Single cycle נקבל:

$$troughput_{single\ cycle} = \frac{n}{6ns \cdot n} = 0.166GHz$$

עבור Multicycle נקבל:

$$troughput_{Multicycle} = \frac{n}{1.25ns \cdot 4 \cdot n} = 0.2GHz$$

עבור ה-Pipelined RISC-V נשים לב כי נתון כי n גדול מאוד ולכן זמן ההגעה של הפקודה הראשונה אל שלב ה-WB זניח. נקבל כי:

$$troughput_{pipeline} = \frac{n}{1ns \cdot n + 4ns} \xrightarrow{n \gg 4} 1GHz$$

שאלה 3 (5 נקודות)

נתונים שני המודולים הבאים:

```
module my_ff (
    input logic clk,
    input logic reset,
    input logic a,
    output logic out,
    output logic outn
);

    always_ff @(posedge clk, posedge reset) begin
        if(reset == 1'b1)
            out <= 1'b0;
        else
            out <= a;
    end

    assign outn = ~out;

endmodule

module my_div (
    input logic in,
    input logic reset,
    output logic out1,
    output logic out2
);

    logic w1, w2;
    my_ff inst1 (in, reset, w1, out1, w1);
    my_ff inst2 (w1, reset, w2, out2, w2);

endmodule
```

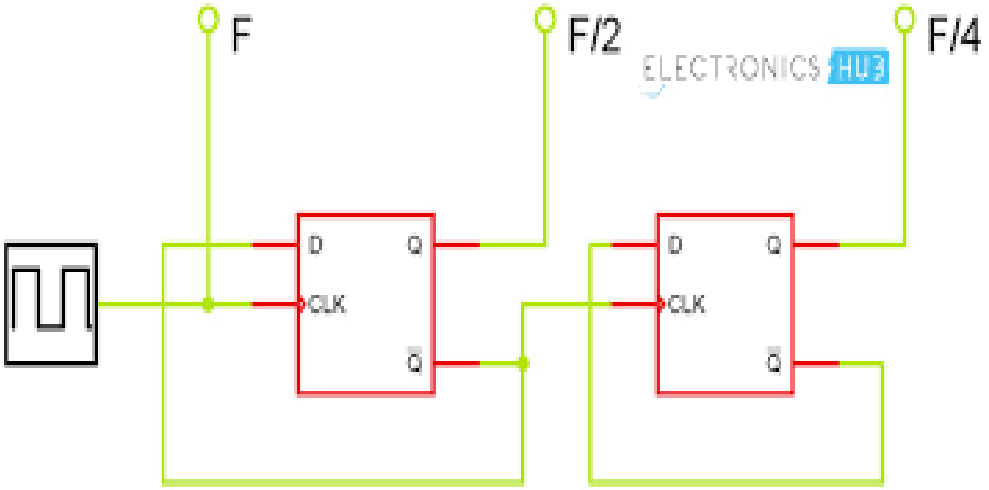
אל כניסת in של המודול my_div מחברים שעון עם תדר f, ולכניסת reset מחברים סיגנל reset. מבצעים אתחול למערכת על ידי העלאה והורדה של סיגנל reset, כאשר לאחר מכן ערכו של סיגנל זה קבוע ושווה ל-0. מה יהיו התדרים של היציאות out1,out2 לאחר התייצבות המערכת?

א. out1 יהיה בתדר f, וout2 יהיה בתדר f/2
ב. out1 יהיה בתדר f/2, וout2 יקבל ערך קבוע
ג. out1 יהיה בתדר f/2, וout2 יהיה בתדר f/4
ד. out1 יהיה בתדר f/4, וout2 יהיה בתדר f/2
ה. שניהם יהיו בתדר f/2

פתרון:

תשובה ג'.

הקוד מממש מחלק תדר בסיסי על ידי אינסטנטציה של שני פליפ-פלופים.



שאלה 4 (5 נקודות)

נתונות שתי מכונות מצבים מסוג MOORE עם כניסה אחת ויציאה אחת. המערכות מתייחסות לרצף הסיביות שנקלטו עד כה כמספר בינארי X (למשל, אם סדרת הקלט הינה 0011, אז X=3).

מכונה א' - מתייחסת לסיבית הנקלטת בכל מחזור כ- **LSB** של המספר X.

מכונה ב' - מתייחסת לסיבית הנקלטת בכל מחזור כ- **MSB** של המספר X.

המכונות יוציאו '1' אם ורק אם המספר X mod 8 הינו **ריבוע** של מספר חיובי שלם כלשהו.
למשל, המספרים 1, 4, 9, 16.... הינם ריבוע של מספר חיובי שלם.
שימו לב: המספר 0 איננו ריבוע של מספר חיובי.

נסמן את מספר המצבים **המינימלי** למימוש מכונה א' בתור S_A, ואת מספר המצבים **המינימלי** למימוש מכונה ב' בתור S_B. סמנו את התשובה הנכונה ביותר:

א. **אחת** מהמכונות **לא** ניתנות למימוש כמכונת מצבים סופית.
ב. **שתי** המכונות **לא** ניתנות למימוש כמכונת מצבים סופית.
ג. שתי המכונות ניתנות למימוש כמכונות מצבים סופיות, ו- $S_A > S_B$.
ד. שתי המכונות ניתנות למימוש כמכונות מצבים סופיות, ו- $S_A = S_B$.
ה. שתי המכונות ניתנות למימוש כמכונות מצבים סופיות, ו- $S_A < S_B$.

פתרון:

תשובה ג'.

עבור מכונה א' – אם המספר שקיבלנו עד כה הוא X_t, אז כשנקבל סיבית חדשה המספר ישתנה ל- $X_{t+1} = 2X_t + b$ כאשר b היא הסיבית החדשה שהתקבלה וערכה 0 או 1. השארית עוברת את אותו תהליך: מוכפלת ב- 2, ומוסיפים לה את הביט שהתקבל (כמובן שאם היא חורגת מ- 8, אז חוזרת ל – 0).

למשל, נניח ומהספר עד כה היה 17, אז 17 mod 8 = 1. אם כעת נקבל 1 כ- LSB, אז המספר החדש הינו 17 * 2 + 1 = 35, שארית החלוקה ב- 8 היא כמובן 3.

ניתן לחזות במדויק מכל מצב לאיזה מצבים נוכל לעבור, ונותר לסמן את המצבים בהם התוצאה תהיה '1'. המצבים הם כמובן מצב שמתאר את השארית 1 ומצב שמתאר את השארית 4.

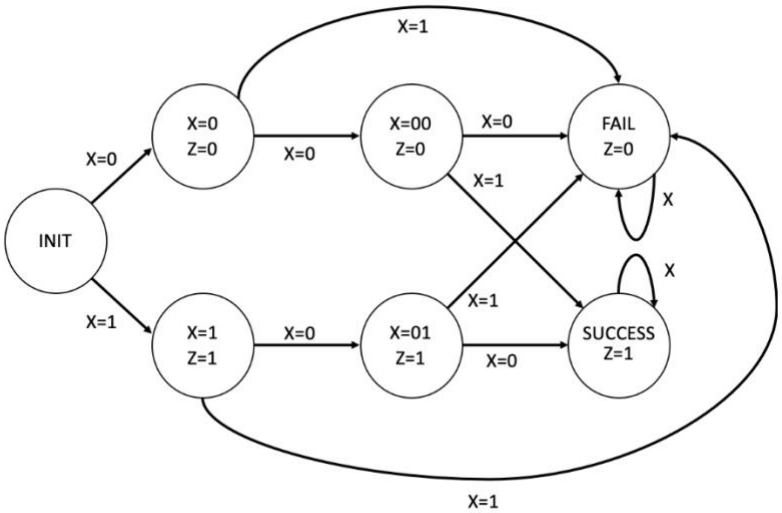
בסה"כ אנחנו נדרש ל- 8 מצבים, כאשר כל מצב יתאר את שארית החלוקה של המספר הנוכחי ב- 8. לא נצטרך מצב התחלתי, אלא נוכל להתחיל מהמצב שבו שארית החלוקה היא 0, משום שהמוצא הוא '0' במצב זה בכל מקרה.

עבור מכונה ב' – החל מסיבית מספר 4, כל סיבית חדשה תגדיל את המספר במספר כלשהו שהוא חזקת 2 והוא גדול מ- 8, ולכן לא תשנה את התוצאה. כלומר, נצטרך לזכור רק את 3 הסיביות הראשונות שיתקבלו לנו, וממצב זה התוצאה לא תשתנה. אם למשל התקבלה הסדרה 001, אז שארית החלוקה ב- 8 היא כמובן 1, ולכן המוצא יהיה 1. לא משנה אילו סיביות נקבל מעתה ואילך, התוצאה תישאר 1.

כעת נשים לב שאם קיבלנו את הסדרה 11 או 10, המשך הסדרה לא משנה ובטוח נוציא '0'. אם קיבלנו כל סדרה שאינה 100 או 001 בקבלת הסיבית השלישית גם כן נוציא '0', אחרת נוציא '1'.

בסה"כ נגדיר שני מצבים FAIL ו- SUCCESS, כך שאם המערכת הגיעה לסדרות הרצויות 100 או 001 היא תעבור ל- SUCCESS, ואם היא הגיעה לסדרה אחרת – 10, 11, 000, 101, אז היא תעבור ל- FAIL (שימו לב שהמצב 10 מכיל את שתי הסדרות 010 ו- 110 ששתיהן כבר יוציאו 1).

מצורפת דיאגרמת המצבים המצומצמת ביותר, לצרכי למידה עתידיים, אך כמובן שניתן לפתור ללא שרטוט הדיאגרמה. ניתן גם להשתמש באלגוריתם צמצום מכונות המצבים כדי להגיע למכונה המצומצמת ביותר.



שאלה 5 (5 נקודות)

נתון משדר עם שעון פנימי העובד בקצב $150MHz$, ומקלט עם שעון פנימי העובד בקצב $30Mhz$.

שני סטודנטים מעוניינים לתקשר ביניהם בעזרת פרוטוקול UART הנלמד בקורס. המקלט מזהה את תחילת סיבית ה-START מיד עם קבלתה. **מבין התשובות הבאות**, מהו קצב השידור ($\frac{1}{T_{bit}}$) המקסימלי האפשרי המאפשר שידור ללא שגיאה?

א. 7.5MHz
ב. 15MHz
ג. 30MHz
ד. 45MHz
ה. אף אחת מן התשובות הנתונות לא תוכל להבטיח שידור תקין

פתרון:

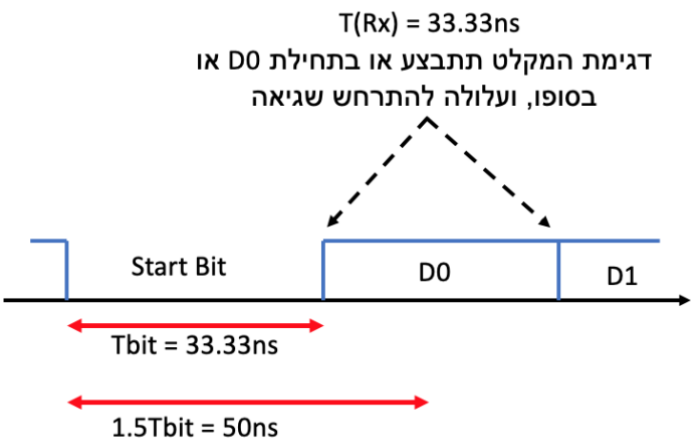
תשובה ב'.

$$T_{cycle}(Tx) = \frac{1}{150MHz} = 6.667ns$$

$$T_{cycle}(Rx) = \frac{1}{30MHz} = 33.333ns$$

כלומר קיבלנו: $5T_{cycle}(Tx) = T_{cycle}(Rx)$.

אם המשדר ישדר באותו קצב של המקלט (30MHz) המקלט לא יוכל לקלוט את 1.5Tbit, אלא יקלוט או ב- Tbit או ב- 2Tbit, ולכן כבר מרגע זה הוא עלול לקלוט שגיאה, כפי שמתואר בשרטוט הבא:



באופן דומה, אם נבחר ב- 45MHz אז $T_{bit} = 22.222ns$. המקלט יוכל לקלוט את 1.5Tbit, אבל מיד אחרי זה תחול שגיאה, *ובאותו אופן אם נבחר ב- 60MHz, אז $T_{bit} = 16.666ns$, והמקלט יוכל לדגום בפעם הראשונה רק בין D0 ל- D1.*

אם נבחר ב- 15MHz, אז $T_{bit} = 66.667ns$, והמקלט יוכל לדגום את הביט הראשון בדיוק ב- 1.5Tbit, ומכאן לדגום כל סיבית ללא שגיאה.

שאלה 6 (5 נקודות)

נתונות הפונקציות הבאות:

$$f(w,x,y,z) = \sum(0,2,3,5,7,8,9,12,13)$$

$$g(w,x,y,z) = \prod(0,1,5,6,10,11,12)$$

נגדיר את הפונקציה $h(w,x,y,z)$:

$$h(w,x,y,z) = nand\ (f(w,x,y,z), g(w,x,y,z))$$

תזכורת: $nand(a,b) = \overline{(a \cdot b)}$

כאשר מצמצמים את $h(w,x,y,z)$ כסכום מכפלות, אילו מהביטויים הבאים יכולים לתאר את הפונקציה המצומצמת?

א. $h(w,x,y,z) = y + z + w'x + x'w$
ב. $h(w,x,y,z) = w'y'z' + w'xy' + wxy + wxz'$
ג. $h(w,x,y,z) = wx'y' + wy'z + w'yz + w'x'z$
ד. $h(w,x,y,z) = w'y' + xz' + wy$
ה. $h(w,x,y,z) = x'y' + xy$

פתרון:

תשובה ד'.

ראשית נמיר את $g(w,x,y,z)$ לסכום מכפלות, נקבל:

$$g(w,x,y,z) = \sum(2,3,4,7,8,9,13,14,15)$$

$$f(w,x,y,z) = \sum(0,2,3,5,7,8,9,12,13)$$

כעת נחשב את $h(w,x,y,z)$ איבר איבר ונקבל:

$$h(w,x,y,z) = \sum(1,3,4,5,6,10,11,12,14,15)$$

נציב את הביטוי למפת קרנו ונקבל:

wx yz	00	01	11	10
00	1	1	1	
01	1	1		
11			1	1
10		1	1	1

נקבל לאחר הצמצום: $h(w,x,y,z) = w'y' + xz' + wy$

תשובה א' נכונה במידה והתייחסתם ל g בתור סכום מכפלות.

תשובה ב' נכונה במידה והתבלבלתם בין פונקציית nand לפונקציית XOR.

תשובה ג' נכונה במידה וביצעתם and במקום nand

שאלה 7 (5 נקודות)

בחברת עורכי הדין "ייצוג הולם" משתמשים בשיטה נקודה צפה על מנת לייצג מספרים. לאחרונה הבין המנהל בחברה כי הם לא צריכים לייצג טווח גדול של מספרים ומעדיפים להתמקד ברזולוציה של המספרים (רזולוציה מוגדרת כהפרש בין הייצוג של שני מספרים עוקבים).

סמנו את האפשרות אשר תיתן את הרזולוציה הטובה ביותר בעדיפות עליונה, ושימוש במספר הביטים המינימלי האפשרי בעדיפות שניה.

מה היית מציע/ה למנהל לשנות?

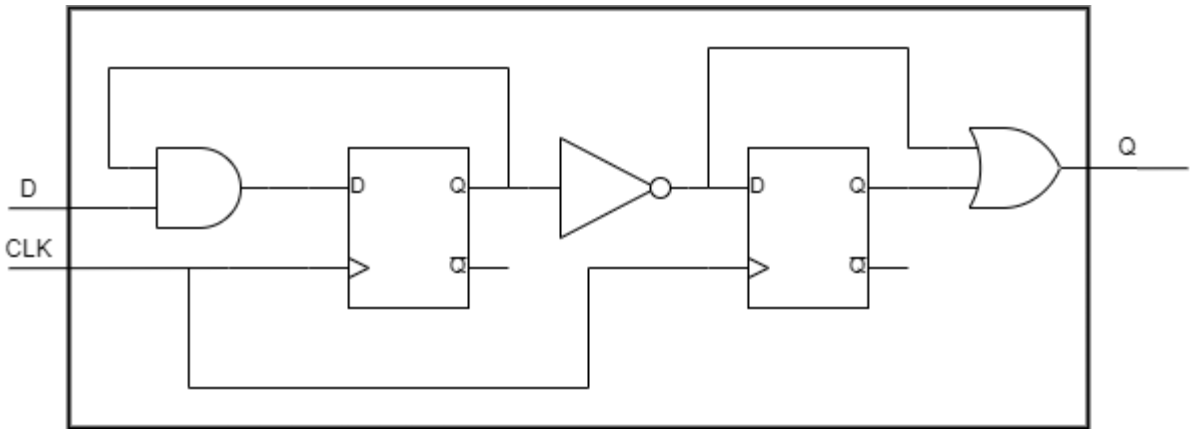
א. להגדיל את ה- exponent באמצעות הוספת ביטים.
ב. להגדיל את שדה ה- fraction באמצעות הוספת ביטים.
ג. להגדיל את ה- exponent על חשבון שדה ה- fraction.
ד. להגדיל את ה- fraction על חשבון שדה ה- exponent.
ה. לעבור לשיטת ייצוג מספרים שלמים.

פתרון:

תשובה ד'.
אנחנו יודעים כי שדה ה-exponent קובע את טווח המספרים שאנחנו יכולים לייצג בזמן ששדה הfraction מגדיר את הרזולוציה בין שני מספרים קרובים.
מכיוון שהרזולוציה מהווה גורם מגביל, וכי טווח המספרים שלנו לא גדול, נוכל להגדיל את שדה הfraction על חשבון שדה הexponent ובכך להגדיל דיוק אך להקטין טווח.
גם התשובה של להגדיל את שדה הfraction/ exponent בהתאמה באמצעות הגדלת מספר הביטים, היא נכונה. אך אינה הנכונה ביותר מכיוון שדורשת הוספת ביטים מיותרים.
אנחנו יודעים כי אלו ביטים מיותרים בגלל שהודגש כי במקרה אחד הטווח שלנו קטן ובמקרה השני הדיוק אינו חשוב.

שאלה 8 (5 נקודות)

נתונה המערכת הבאה:



את המערכת יש לשווק כקופסא שחורה, כלומר מי שיקנה את הרכיב לא ידע על המימוש הפנימי שלו, אלא ידע שיש 2 כניסות (כניסת שעון וכניסת מידע) ויציאה אחת.

נתון: (הזמנים ב-ns)

	t_{cd}	t_{pd}/t_{pCQ}	t_{setup}	t_{hold}
AND	6	10		
OR	6	10		
FF	5	10	20	10
NOT	5	6		

נתון כי ערכו של מחזור השעון הוא $T_{clk} = 40ns$.

בתחילת פעולתו של המעגל ערכה של הכניסה שווה ל-1 למשך פרק זמן ארוך, כלומר ערכה של היציאה הוא 0.

ערך הכניסה משתנה מ-1 ל-0 תוך שמירה על תנאי ה-setup וה-hold של המערכת כולה.

בהינתן נתוני הרכיבים, מהו פרק הזמן אשר חולף מעליית השעון עד להתייצבות היציאה של המערכת על ערכה החדש? ניתן להזניח את תנאי ה-hold.

א. $26ns$
ב. $36ns$
ג. $46ns$
ד. $60ns$
ה. $90ns$

פתרון:

ערך הכניסה יורד מ1 ל-0. מיד לאחר $t_{pd}(and)$ יציאת שער ה-*AND* יורד ל-0 (מכיוון שזה שער *AND* מספיק שרגל אחת שלו תהיה 0 כדי שהיציאה תשתנה).

לאחר מכן, לפי נתוני השאלה, הכניסה ל*FF* יציבה מספיק זמן לפני עליית השעון ומקיימת את תנאי המשטר הדינאמי.

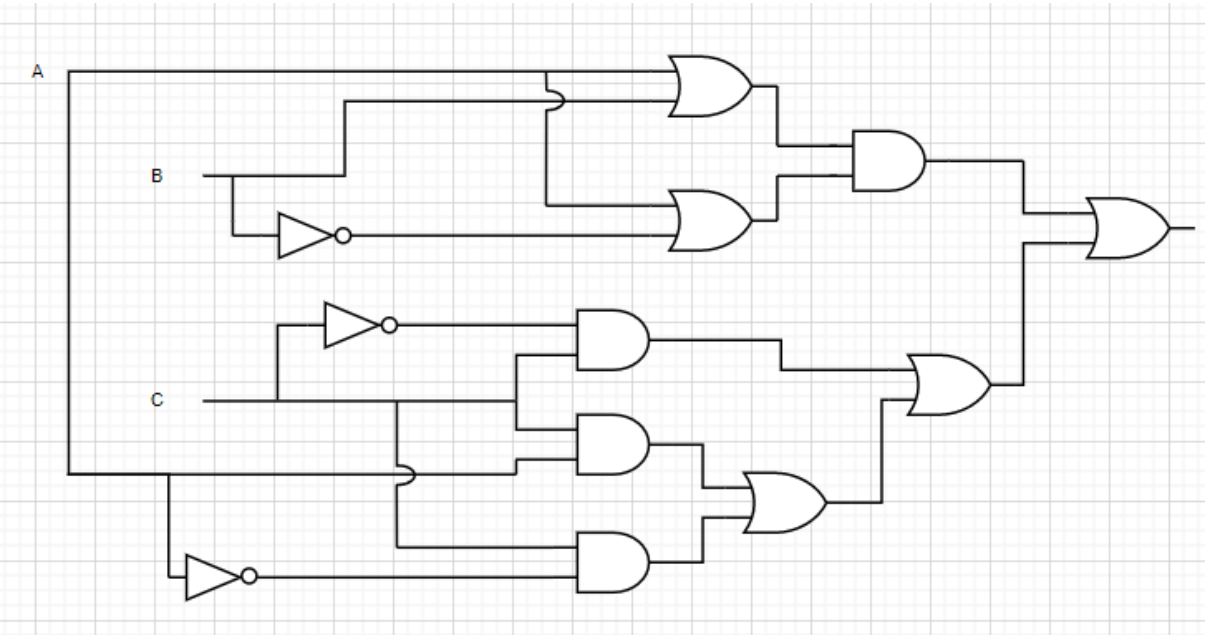
t_{pCQ} לאחר עליית השעון, יציאת ה*FF* מתעדכנת ל-0. $t_{pd}(NOT)$ לאחר מכן, יציאת שער ה*NOT* עולה ל1. מכיוון שיציאת שער ה*NOT* מחובר לשער *OR*, יציאת המערכת מתעדכנת $t_{pd}(OR)$ מרגע זה (זאת בגלל שמדובר בשער *OR* ומספיק שרק רגל אחת תעלה ל-1 כדי שיציאת השער תתעדכן).

כלומר סה"כ הזמן שעובר מעליית השעון:

$$T = t_{pCQ}(FF) + t_{pd}(NOT) + t_{pd}(OR) = 26ns$$

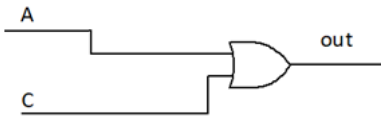
שאלה 9 (5 נקודות)

נתון המעגל הצירופי הבא:

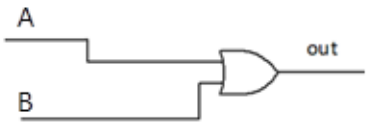


מה מבין האפשרויות הבאות היא המימוש המצומצם ביותר של המעגל הצירופי הנתון?

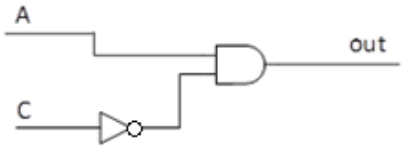
א.



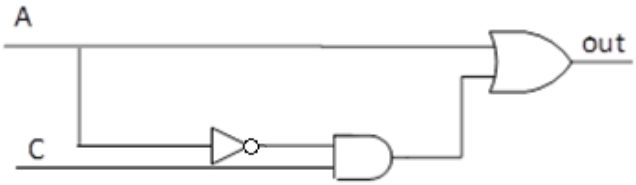
ב.



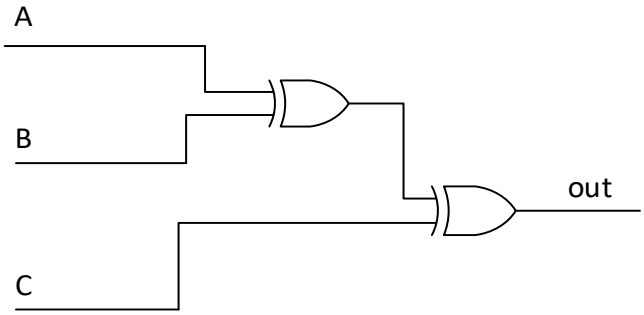
ג.



ד.



ה.



פתרון:

תשובה נכונה א'.

הפונקציה הממשת את המעגל היא:

$$(A+B)(A+\bar{B})+AC+\bar{A}C+C\bar{C}=$$

$$=(A+B\bar{B})+C(A+\bar{A})=A+C$$

שאלה 10 (5 נקודות)

במעבד Multicycle RISC-V המצורף בנספח הבחינה מעוניינים להוסיף תמיכה בחריגת גלישה (overflow), אשר מוגדרת כגלישה ב-ALU אשר יכולה להתרחש **בכל שימוש של ה-ALU.** הבקר ממומש באמצעות Dispatch ROM כפי שנלמד בקורס.
על מנת לתמוך בחריגת הגלישה ניתן לשנות Dispatch קיים או להוסיף חדש בעת הצורך.

כמה Dispatch ROM חדשים יש להוסיף? לא ניתן לשנות את מספרי המצבים.

א. אין צורך בהוספת Dispatch נוספים.

ב. יש צורך להוסיף Dispatch יחיד.

ג. יש צורך להוסיף שני Dispatch.

ד. יש צורך להוסיף ארבעה Dispatch.

ה. יש צורך להוסיף שישה Dispatch.

פתרון:

יש להוסיף יציאה מכל אחד מהמצבים 0,1,2,6,8,9 עבור מצבים 1,2 יש להוסיף שורות ב- dispatch.
עבור מצבים 0,6,8,9 יש להוסיף dispatch חדשים. סה"כ יש להוסיף 4 dispatch חדשים.

שאלה 11 (5 נקודות)

עובד מוכשר הציע להחליף את רכיב הזיכרון במעבד Multicycle RISC-V ביחידת זיכרון שיצר בעצמו. יחידת הזיכרון היא בעלת 3 כניסות מידע - offset, data, address (בנוסף לכניסת Read/Write). יחידת הזיכרון החדשה מסוגלת לחשב בעצמה את המיקום בזיכרון אליו ניגשים.

בנוסף נתונים זמני ההשהיה של שלבי הביצוע השונים של המעבד (לפני השינוי):

שלב	*WB*	*MEM*	*EXE*	*DECODE*	*FETCH*
זמן השהיה	$70ns$	$90ns$	$110ns$	$80ns$	$50ns$

זמן ההשהיה של *MEM* עם יחידת הזיכרון החדשה הוא $120ns$.

ניתן להניח שנעשו כל השינויים הרלוונטיים במעבד כדי לתמוך בהחלפה זו ושהשינוי היחיד שהתבצע מבחינת זמני ההשהיה נובע משינוי רכיב הזיכרון בלבד.

מנהלו של העובד רוצה להיעזר בכם בכדי להחליט האם כדאי לו להחליף את רכיב הזיכרון במעבד, מטרתו של המנהל היא להריץ את התוכנית מהר ככל הניתן. התוכנית מורכבת מהרכב הפקודות הבא:

פקודה	*add*	*beq*	*lw*	*sw*	*subi*
יחס	1.5β	0.35	β	0.15	שאר הפקודות

האם כדאי למנהל להחליף את רכיב הזיכרון (עליכם לבחור את התשובה הנכונה בעלת ערך β קטן ככל הניתן)?

א. המעבד הישן עדיף בכל מקרה

ב. המעבד החדש עדיף לכל $\beta > 0.15$

ג. המעבד החדש עדיף לכל $\beta > 0.17$

ד. המעבד החדש עדיף לכל $\beta > 0.19$

ה. המעבד החדש עדיף בכל מקרה

פתרון:

נשים לב שזמן המחזור של המעבד הישן (תלוי בחלק הארוך ביותר) הינו $110ns$, *זמן המחזור של המעבד החדש הינו* $120ns$. *עם זאת נשים לב שעבור פקודות LW ו SW אנו זקוקים למחזור שעון אחד פחות בלבד (לעומת המעבד הישן).*

עבור פקודות add אנו משתמשים ב-4 מחזורי שעון, עבור beq משתמשים ב 3 מחזורי שעון, עבור lw אנו משמשים ב-5 (במעבד הישן) ועבור sw אנו משתמשים ב 4 (במעבד הישן).

זמן הביצוע של התוכנית בשימוש במעבד הראשון הינו:

$$Total\ old\ time = \frac{(4 \cdot 1.5\beta + 35 \cdot 3 + 5 \cdot \beta + 4 \cdot 15 + 4 \cdot (50 - 2.5\beta))}{100} \cdot 110ns =$$

$$= 66\beta + 55\beta + 1155ns + 660ns + 2200ns - 110ns = 4015ns + 11\beta$$

$$Total\ new\ time = \frac{\left(4 \cdot 1.5\beta + 35 \cdot 3 + 4 \cdot \beta + 3 \cdot 15 + 4 \cdot (50 - 2.5\beta)\right)}{100} \cdot 120n =$$

$$= 72\beta + 1260ns + 48\beta + 540ns + 2400ns - 120\beta = 4200ns$$

נבדוק מתי המעבד החדש טוב יותר מהמעבד הישן:

$$Total\ new\ time < Total\ old\ time$$

$$4200ns < 4015ns + 11\beta$$

$$185 < 11\beta$$

$$16.818 < \beta$$

לכן המעבד החדש יותר יעיל מהמעבד הישן לכל $\beta > 0.16818$.

שאלה 12 (10 נקודות)

הערה: שאלה זו דומה לשאלה 11 אך אינה מתבססת על הנתונים משאלה 11

עובד מוכשר הציע להחליף את רכיב הזיכרון במעבד RISC-V **Single Cycle** ביחידת זיכרון שיצר בעצמו. יחידת הזיכרון היא בעלת 3 כניסות מידע - offset, data, address (בנוסף לכניסת Read/Write). יחידת הזיכרון החדשה מסוגלת לחשב בעצמה את המיקום בזיכרון אליו ניגשים.

מנהלו של העובד רצה להחליט האם כדאי להשתמש ברכיב הזיכרון החדש. הניחו כי המנהל רוצה להפחית את זמן המחזור של המעבד בעדיפות עליונה.

נתונים זמני ההשהיה של חלקי המעבד השונים (לפני השינוי):

שלב	*WB*	*MEM*	*EXE*	*DECODE*	*FETCH*
זמן השהיה	$70ns$	$90ns$	$110ns$	$80ns$	$50ns$

יחידת הזיכרון החדשה מסוגלת לחשב בעצמה את מיקום הגישה לזיכרון וזמן ההשהיה של השלב לאחר ההחלפה הוא $120ns$.

ניתן להניח שנעשו כל השינויים הרלוונטיים במעבד כדי לתמוך בהחלפה זו ושהשינוי היחיד שהתבצע מבחינת זמני ההשהיה נובע משינוי רכיב הזיכרון בלבד.

א. מה זמן המחזור של המעבד החדש? _____

ב. בהינתן כי אנו מריצים תוכנית שאינה כוללת פקודות גישה לזיכרון (אך על המעבד לתמוך בפקודות אלו). מהו היחס בין זמן הביצוע של התוכנית במעבד החדש לעומת זמן הביצוע של התוכנית במעבד הישן?

$$\frac{T_{new}}{T_{old}} =$$

פתרון:

נשים לב שזמן המחזור של המעבד הישן כנלמד בכיתה הוא:

$$T = t_{pd}(FETCH) + t_{pd}(DECODE) + t_{pd}(EXE) + t_{pd}(MEM) + t_{pd}(WB) =$$

$$= 50 + 80 + 110 + 90 + 70 = 400ns$$

לאחר השינוי עלינו לבדוק מהי הפקודה הארוכה ביותר, במעבד רגיל הפקודה הארוכה ביותר הינה פקודת LW, נבדוק את אורכה (נשים לב שאין צורך ב *ALU* כי את חישוב הכתובת ניתן לבצע ברכיב הזיכרון).

$$T = t_{pd}(FETCH) + t_{pd}(DECODE) + t_{pd}(NEW\ MEM) + t_{pd}(WB) =$$

$$= 50 + 80 + 120 + 70 = 320ns$$

בנוסף עלינו לבדוק פקודה מסוג *R-type* (אינה כוללת זיכרון) ומכילה את שאר השלבים, נקבל:

$$T = t_{pd}(FETCH) + t_{pd}(DECODE) + t_{pd}(EXE) + t_{pd}(WB) =$$

$$= 50 + 80 + 110 + 70 = 310ns$$

כלומר זמן המחזור של המעבד עם הרכיב החדש הינו $T = 320ns$.

נניח שבתוכנית ישנן α פקודות, זמן הריצה של התוכנית על כל מעבד הוא:

$T_{prog} = \alpha T$. נחשב את היחס ונקבל:

$$\frac{T_{new}}{T_{old}} = \frac{\alpha T_{new}}{\alpha T_{old}} = \frac{T_{new}}{T_{old}} = \frac{320}{400} = 0.8$$

שאלה 13 (13 נקודות)

נתון מעבד מסוג PIPELINE RISC V. נתוני המעבד:

- *מכיל Bypass בתוך ה- Register File (כלומר, Forwarding בין שלב ה- WB ל- Decode).*
- ***אינו** מכיל Forwarding נוסף.*
- **אינו** מכיל *Load Hazard Detection Unit* (כלומר לא ניתן לעצור את הצינור בשלב ה-*Decode* במקרה של *Lw hazard*).

איור ה- *Datapath* של המעבד הינו כפי שנלמד, ונמצא בדפי העזר.

סטודנט בקורס מעוניין להעביר מילה בזיכרון מכתובת *0x200* לכתובת *0x400*. לצורך כך הוא מימש את הקוד הבא:

```
1   addi   t0, x0, 0x200
2   addi   t1, x0, 0x400
3   lw     t2, 0(t0)
4   sw     t2, 0(t1)
```

נתון כי זמני ההשהייה של השלבים השונים נתונים בטבלה הבאה:

Fetch	***Decode***	***Execute***	***Memory***	***Writeback***
300ns	*50ns*	*150ns*	*300ns*	*50ns*

זמני פעולת הרגיסטרים בין השלבים זניחים.

סעיף א'

מהנדס זוטר בחברה, העיר לסטודנט שהקוד לא יוכל לרוץ כהלכה משום שלדעתו יתרחש Hazard. המהנדס הציע להוסיף פקודות NOP על מנת שהמעבד יוכל להריץ את הקוד.
עליכם למלא בטבלה הבאה את כמות פקודות ה- *NOP* **המינימאלית** הנדרשת לריצה תקינה של הקוד על המעבד הנתון. אם לדעתכם אין צורך להוסיף פקודות *NOP* כלל, מלאו 0 בשורה הראשונה בטבלה (טבלה שתישאר ריקה תחשב כשאלה שלא נענתה).

לדוגמה: אם לדעתכם צריך להוסיף 3 פקודות *NOP* בין פקודות 5 ל- 6 אז תמלאו באופן הבא:

כמה *NOP* להוסיף	אחרי פקודה מספר	לפני פקודה מספר
3	5	6

מלאו או הטבלה הבאה (אין הכרח למלא את כולה):

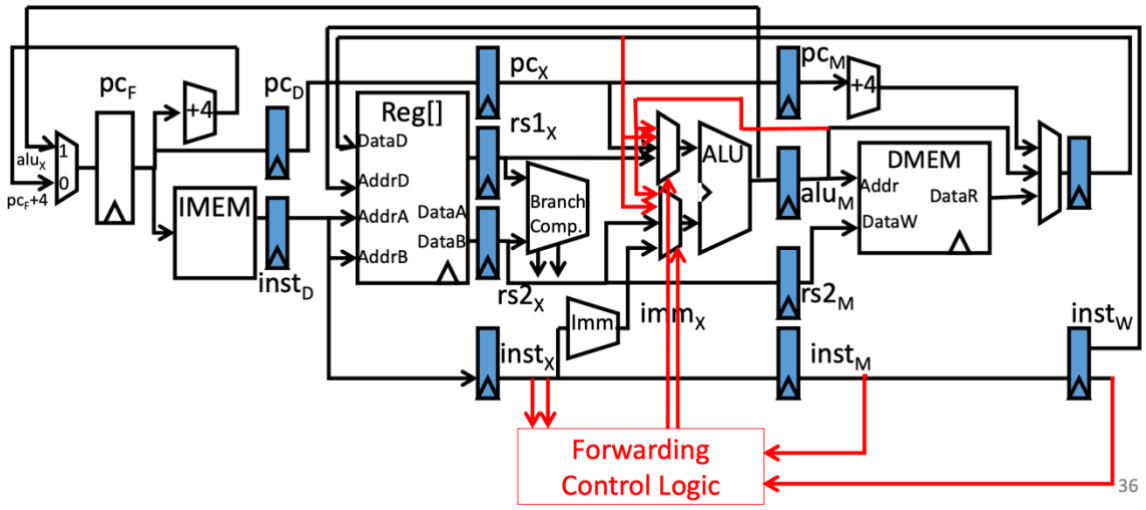
כמה *NOP* להוסיף	אחרי פקודה מספר	לפני פקודה מספר

סעיף ב'

בהנחה שנוספו בסעיף הקודם *X* פקודות *NOP*, מה יהיה זמן ריצת הקוד כפונקציה של *X*?
מלאו במלבן להלן את החישוב הדרוש.
שימו לב שזמן הריצה מוגדר מתחילת ריצת הפקודה הראשונה ועד שהפקודה האחרונה (פקודת ה-*SW*) יוצאת מה- *Pipe*. ניתן להניח שלאחר פקודת ה- *SW* ירוצו פקודות *NOP* לפי הצורך.

סעיף ג'

כעת מהנדס בכיר בחברה החליט להוסיף יחידת *Forwarding Unit* במטרה להפחית את כמות ה- *NOP* בקוד כפי שמופיע באיור להלן. יחידה זו מטפלת ב- *Forwarding* משלב ה- *Memory* ומשלב ה- *Writeback* לשלב ה- *Execute*, וספציפית לבוררים שבכניסה ל- *ALU*.



מלאו בטבלה להלן איך תשתנה כמות פקודות ה- NOP המינימאלית הנדרשת לריצה תקינה של הקוד על המעבד הנתון ***לאחר השיפורים שנוספו****. אם לדעתכם אין צורך להוסיף פקודות NOP כלל, מלאו 0 בשורה הראשונה בטבלה (טבלה שתישאר ריקה לא תיבדק).*

כמה *NOP* להוסיף	אחרי פקודה מספר	לפני פקודה מספר

פתרון:

סעיף א':

```
1   addi   t0, x0, 0x200
    NOP
2   addi   t1, x0, 0x400
3   lw     t2, 0(t0)
    NOP
    NOP
4   sw     t2, 0(t1)
```

בין פקודות 3 ל- 4 נהיה חייבים להוסיף 2 פקודות *NOP*, בגלל שה- *LW* כותב לרגיסטר *t2*, וה- *sw* קורא ממנו את הערך הלא מעודכן. יש *Forwarding* משלב ה- *WB* לשלב ה- *Decode* אז 2 פקודות *NOP* יספיקו.

בנוסף, יש *Hazard* בין פקודות 1 ו- 3. אז נצטרך להוסיף *NOP* נוסף או בין פקודות 1 ו- 2, או בין פקודות 2 ו- 3. בסה"כ 3 פקודות *NOP* נוספו. שימו לב שבכל מקרה לא יהיה *Hazard* בין פקודות 2 ו- 4.

סעיף ב':

סה"כ נריץ 4 פקודות + *X* פקודות *NOP* (אם פתרתם נכון את סעיף א' – אז *X=3*, אך השאלה נבדקה לפי כמות ה- *NOP* שהוספתם). זמן המחזור של המעבד נקבע ע"י השלב האיטי ביותר שהינו *300ns*.

במהלך 4 המחזורים הראשונים ה- *Pipe* יתמלא, ולאחר מכן בכל מחזור תסיים פקודה אחת. בסה"כ זמן הריצה יהיה $4 \cdot 300ns + (4 + x) \cdot 300ns$.

סעיף ג':

עדיין יש להוסיף 2 פקודות *NOP* בין פקודה 3 ל- 4. זאת משום שאין למעבד *Forwarding* לחלק ה- *DATA* שבשלב ה- *EXECUTE* (הקו שיוצא מרגיסטר *rs2*) אלא רק ל- *ALU*. כלומר, אם היינו צריכים לעשות *Forwarding* מהערך החדש של *t2* לרגיסטר שמשתמשים בו לכתובת בפקודת ה-*sw* אז היינו צריכים להוסיף רק פקודת *NOP* אחת, אבל במקרה שלנו אנחנו רוצים להעביר את הערך של *t2* ל- *DATA* של פקודת ה- *SW* ולכן נצטרך שתי פקודות *NOP*.

שאלה 14 (10 נקודות)

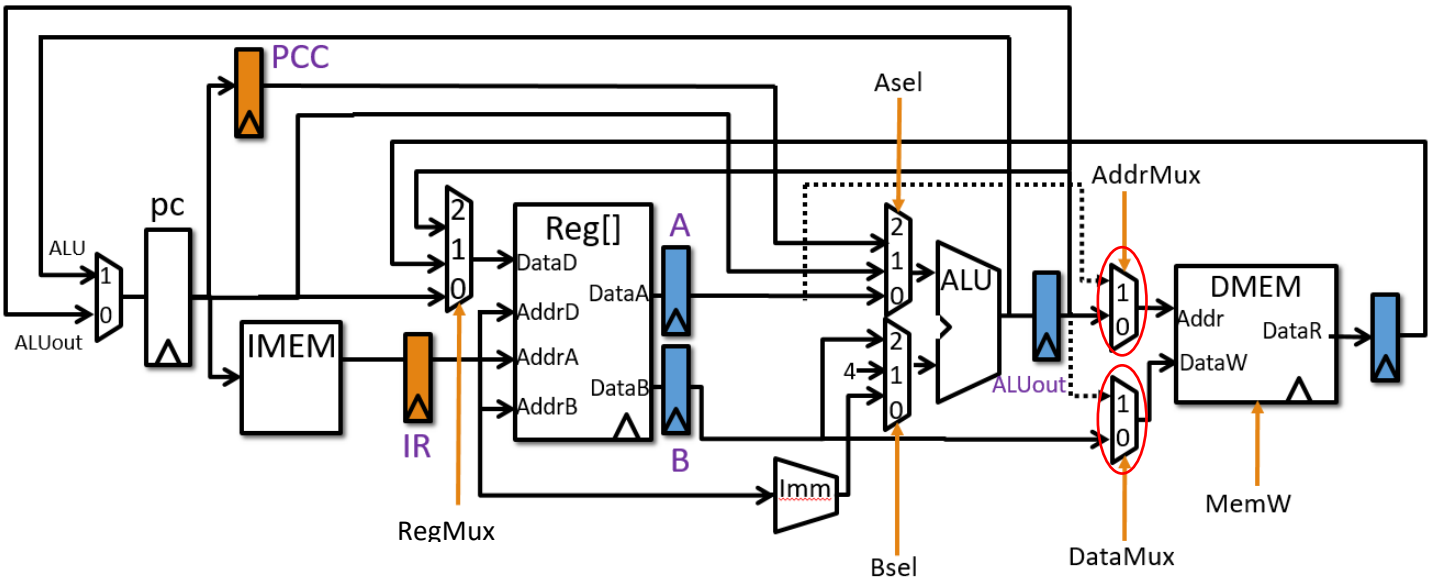
המהנדס הנודע ארכי טקט מעוניין להוסיף תמיכה לארכיטקטורת ה-Multicycle RISC-V בפקודה החדשה הבאה:

$StorePC\ rs1, imm$

פקודה זו משתמשת בכתובת הזיכרון אשר שמורה ברגיסטר $rs1$, ומאחסנת בכתובת זו את הערך $PC + imm$ ($mem[Reg[rs1]] = PC + imm$).

הניחו כי פורמט הפקודה הוא S-type.

לצורך התמיכה בפקודה הוסיף המהנדס שני בוררים (מוקפים במעגל) ומספר חיווטים (קווי מידע חדשים מסומנים עם קו מקווקו).



מלאו את הטבלה הבאה בערכי הסיגנלים המצוינים, כך שפעולת הסיגנלים בכל מחזור שעון תתאים למימוש פקודת ה-$StorePc$. יש למלא את הטבלה כך שמספר מחזורי השעון המממשים את הפקודה הוא **מינימלי** (עבור המימוש הנתון). לא כל עמודות הטבלה נדרשות בהכרח. כאשר ערכו של סיגנל אינו רלוונטי במחזור שעון מסוים, יש לסמן את הערך Φ (don't care).

מס' מחזור שעון	1	2	3	4	5	6	7	8
$Asel$								
$Bsel$								
$AddrMux$								
$DataMux$								
$MemW$								

פתרון:

נשים לב כי למרות שערכי הסינגלים בטבלה זהים במחזורים 2 ו-3, ערך ה-imm שונה ולכן פעולת המחזור שונה. כלומר, אנו זקוקים ל-4 מחזורי שעון (בדומה לפעולת sw רגילה).

מס' מחזור שעון	IF	DEC	Ex	MEM
$Asel$	1	2	2	Φ
$Bsel$	1	0	0	Φ
$AddrMux$	Φ	Φ	Φ	1
$DataMux$	Φ	Φ	Φ	1
$MemW$	0	0	0	1

שאלה 15 (7 נקודות)

נתון מעבד pipelined RISC-V כפי שנלמד בכיתה בעל יחידת hazard detection unit כמו שנלמד בקורס ובעל ה-forwarding הבאים: WB-DE, WB-EXE, MEM-EXE.

ההכרעה על קפיצה מתרחשת בשלב ה-EXE והמעבד מניח תמיד שהקפיצה לא נלקחת ומבצע flush במידה וכן.

נתון קוד האסמבלי הבא:

הניחו כי $s0$ הינו מצביע לתחילת מערך ידוע כלשהו.

	$0x00001000$	$lw\ t0, 0(s0)$
	$0x00001004$	$addi\ t0, t0, 4$
	$0x00001008$	$add\ t0, t0, t0$
	$0x0000100C$	$lw\ t1, 4(s0)$
	$0x00001010$	$addi\ t1, t1, 4$
	$0x00001014$	$add\ t1, t1, t1$
	$0x00001018$	$addi\ t2, x0, 5$
Loop:	$0x0000101C$	$lw\ t3, 8(s0)$
	$0x00001020$	$add\ t4, t3, t0$
	$0x00001024$	$add\ t4, t4, t1$
	$0x00001028$	$subi\ t2, t2, 1$
	$0x0000102C$	$bne\ t2, x0, loop$
	$0x00001030$	NOP
	$0x00001034$	NOP
	$0x00001038$	NOP

מרגע כניסת הפקודה הראשונה לשלב ה-*Fetch*, כמה מחזורי שעון יעברו עד לסיום התוכנית (כלומר, עד ליציאת הפקודה בשורה $0x0000102C$ בפעם האחרונה משלב ה*WB*). יש לפרט את החישוב בקצרה (לא יותר משורה אחת).

פתרון:

נשים לב כי יש לנו 3 מקומות בהם מתרחש *load hazard*, כשבפעם השלישית הוא מתרחש בכל איטארציה של הלולאה כלומר 5 פעמים. סה"כ 7 מחזורי שעון מתבזבזים על *data hazard*.

בנוסף, בכל קפיצה של הלולאה, אנחנו מניחים שהקפיצה לא נלקחת וטוענים 2 *NOP*-ים עד שאנחנו מגלים שטעינו. המצב הזה קורה 4 פעמים ואנחנו למעשה מכניסים 8 בועות בגלל הקפיצות האלו.
סה"כ אנחנו צריכים להוסיף לחישוב שלנו 15 מחזורי שעון שמתבזבזים על קפיצות ועל *data hazards*.

סה"כ יש לנו 7 + 5*5 = 32 פקודות.

ולכן,

$$total\ cycles = 32 + 15 + 4 = 51\ cycles$$

שאלה 16 (10 נקודות)

סטודנט חרוץ החליט לממש אלגוריתם חיפוש בינארי בעזרת קוד אסמבלי. האלגוריתם יבצע חיפוש של איבר מסוים על מערך ממוין של מספרים, כאשר **כל מספר הוא בגודל של 4 בתים**. עקב תקלה, חלקים מן המימוש נמחקו. עליכם להשלים את חלקי הקוד החסרים (מסומנים בקו תחתון) בכדי שהמימוש יפעל כנדרש.

לנוחיותכם מצורף מימוש אלגוריתם החיפוש בקוד c:

```
int binarySearch(int arr[], int l, int r, int x)
{
    while (l <= r) {
        int m = (l + (r - 1)) / 2;
          // Check if x is present at mid
        if (arr[m] == x)
            return m;
          // If x greater, ignore left half
        if (arr[m] < x)
            l = m + 1;
          // If x is smaller, ignore right half
        else
            r = m - 1;
    }
    return -1;
}
```

במהלך המימוש הניחו כי המיפוי בין רגיסטרים למשתנים הוא:

$s0 \rightarrow arr, s1 \rightarrow L, s2 \rightarrow R, s3 \rightarrow X, s4 \rightarrow return\ value$

<u>יש להשלים את הקוד אשר נתון להלן במקומות הנדרשים</u>:

```
0x1AA0 0000                 addi s4, x0, -1          //ret=-1
0x1AA0 0004        loop:    blt s2, s1, ret
0x1AA0 0008                 addi t0, s2, -1
0x1AA0 000C                 add t0, t0, s1
0x1AA0 0010                 srli t3, t0, 1
0x1AA0 0014                 add t0, t3, t3
0x1AA0 0018                 add t0, t0, t0
0x1AA0 001C                 add t2, s0, _______
0x1AA0 0020                 lw t1, 0(t2)
0x1AA0 0024                 beq t1, s3 ________

0x1AA0 0028                 blt t1, ____, if_label
0x1AA0 002C                 addi s2, t3, -1
0x1AA0 0030                 J _______

0x1AA0 0034        if_label: addi s1, t3, 1
0x1AA0 0038                 J loop

0x1AA0 003C        found_x: add s4, x0, _____
0x1AA0 0040            ret:                         // done
```

פתרון:

```
0x1AA0 0000                 addi s4,x0,-1          ret=-1
0x1AA0 0004        loop:    BLT s2, s1, exit       Branch if r<l
0x1AA0 0008                 addi t0, s2, -1        t0=r-1
0x1AA0 000C                 add t0, t0, s1         to=l+r-1
0x1AA0 0010                 srli t3, t0, 1         m=t0/2
0x1AA0 0014                 add t0, t3, t3
0x1AA0 0018                 add t0, t0, t0
0x1AA0 001C                 add t2, s0, t0
0x1AA0 0020                 lw t1, 0(t2)           t1=arr[m]
0x1AA0 0024                 beq t1,s3 found_x

0x1AA0 0028                 blt t1, s3, if_label   If arr[m]<x
0x1AA0 002C                 addi s2, t3, -1        Else, r=m-1
0x1AA0 0030                 J loop

0x1AA0 0034        if_label: addi s1, t3, 1         l=m+1
0x1AA0 0038                 J loop

0x1AA0 003C        found_x: add s4, x0, t3          ret=m
0x1AA0 0040        exit:                            // done
```