מערכות ספרתיות ומבנה המחשב (044252)

סמסטר חורף תשע"ט

בחינה סופית – מועד א

29 בינואר 2019

פתרון

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.

הנחיות והוראות:

- הבחינה כתובה על גבי 21 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.
- רשמו את מספר הסטודנט שלכם על חוברת הבחינה (בראש עמוד זה), על דפי העזר, ועל כל מחברות הטיוטה.
- לא מורדות נקודות (אין "קנס") בגין תשובה שגויה. לכן, כדאי לסמן תשובה כלשהי לכל שאלה.
- ציון השאלות 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).**
- אסור שימוש בכל חומר חיצוני. 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, טלפתית, או אחרת. בפרט, אין להחזיק בטלפון סלולארי וגם לא במחשבון בזמן הבחינה.

בהצלחה!

שאלה 1 (5 נקודות)

נתונות הטענות הבאות:

A. במערכת צירופית המקיימת את המשטר הסטטי מתקיים $Throughput = \frac{1}{Latency}$

B. כדי להגדיל את ה- $Throughput$ **חייבים להקטין** את ה- $Latency$ של המערכת

C. ה-$Latency$ של מערכת מצונרת **בהכרח גדול שווה** ל- $Latency$ של **אותה** מערכת צירופית (בלי הפליפ-פלופים).

תזכורת: המשטר הסטטי קובע שאין לשנות את הכניסה לפני שמוצא המעגל סיים להתעדכן

בחרו את התשובה **הנכונה החזקה ביותר** מבין האפשרויות הבאות:
הערה: תשובה 1 חזקה יותר מ-2 אם נכונותה של 1 גוררת את נכונותה של 2.

א- טענה A נכונה
ב- טענה B נכונה
ג- טענה C נכונה
ד- טענות A ו- B נכונות
ה- טענות A ו- C נכונות

פתרון: תשובה ה'
טענה 1 נכונה – לפי ההגדרה
טענה 2 לא נכונה – במערכת מצונרת אפשר להגדיל את ה-latency (ע"י הוספת עוד stages בצינור) והתפוקה תגדל – ראינו דוגמאות בכיתה.
טענה 3 נכונה – כדי להפוך מערכת צירופית למערכת מצונרת אנחנו מוסיפים פליפ-פלופים, הוספת פליפ-פלופ מוסיפה זמן השהייה וכתוצאה מכך ה-Latency גדל (או נשאר זהה אם הפליפ-פלופים אידאליים).

שאלה 2 (5 נקודות)

נתונים משדר ומקלט העובדים לפי פרוטוקול התקשורת הטורי שנלמד בקורס UART.
המשדר משדר סיבית חדשה כל T_{bit} והמקלט תוכנן בטעות לדגום כל $T_{bit} + \delta$.
המקלט מזהה את Start-Bit מיידית.

כאשר: $\delta \neq 0$, $-T_{bit} < \delta < T_{bit}$

בהנחה שלא יכולות להתרחש שגיאות על קו התקשורת, מבין הערכים הבאים של δ, בחרו את התשובה המאפשרת שידור וקליטה של המספר המירבי של סיביות מידע <u>רצופות</u> באופן תקין?

א- $\frac{T_{bit}}{20}$

ב- $-\frac{T_{bit}}{33}$

ג- $\frac{T_{bit}}{33}$

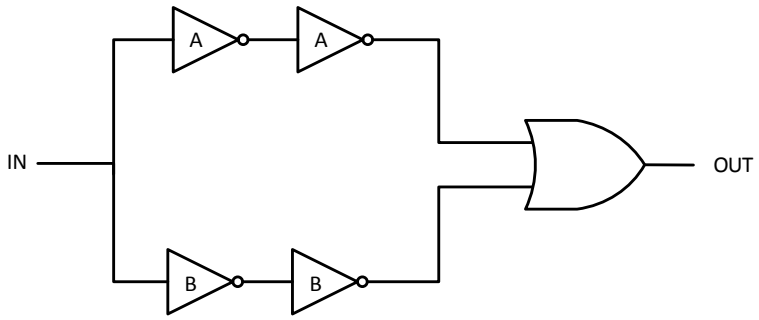
ד- $-\frac{T_{bit}}{64}$

ה- $\frac{T_{bit}}{93}$

פתרון: תשובה ה'
המקלט דוגם בקצב שונה מהקצב בו המשדר שולח מידע (מהיר/איטי יותר, לפי הערך של δ). וכתוצאה מכך, המקלט מתחיל לצבור סטייה ימינה/שמאלה ככל שרצף הביטים ארוך יותר.
במקרה האידאלי, אם $\delta = 0$, המקלט ידגום את הסיביות בדיוק באמצע ולא תהיה סטייה ולכן ניתן לשלוח רצף אינסופי של סיביות מידע. ככל שהסטייה פר ביט גדולה יותר, כך ניתן לשלוח פחות ביטים (לא משנה אם הסטייה היא לצד ימין או לצד שמאל).
ולכן התשובה הנכונה היא ה- $|\delta|$ הקטן ביותר.

שאלה 3 (5 נקודות)

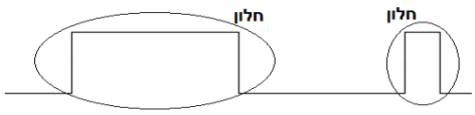
נתונה המערכת הצירופית הבאה



להלן זמני ההשהיה

$t_{pd}(A) = 3\ ns$	$t_{cd}(A) = 2\ ns$
$t_{pd}(B) = 4\ ns$	$t_{cd}(B) = 1\ ns$
$t_{pd}(OR) = 7\ ns$	$t_{cd}(OR) = 5\ ns$

נגדיר "חלון אחדים" כמשך הזמן בו האות נמצא ב- '1' באופן רציף (ברגע שיש '0' החלון מסתיים).



בכניסה IN של המערכת מכניסים חלון אחדים באורך של $25\ ns$. מבין התשובות הבאות, בחרו את **<u>אורכו המירבי האפשרי</u>** של חלון אחדים במוצא של המערכת כתגובה לקלט זה:

א- $27\ ns$
ב- $29\ ns$
ג- $31\ ns$
ד- $33\ ns$
ה- $35\ ns$

פתרון: תשובה ד'

$$
\begin{aligned}
W_{max} &= 25 - T_{min,\uparrow} + T_{max,\downarrow} \\
&= 25 - \left(2 \cdot \min\{t_{cd}(A), t_{cd}(B)\} + t_{cd}(OR)\right) \\
&+ \left(2 \cdot \max\{t_{pd}(A), t_{pd}(B)\} + t_{pd}(OR)\right) \\
&= 25 - (2 \cdot 1 + 5) + (2 \cdot 4 + 7) = 33\ ns
\end{aligned}
$$

שאלה 4 (5 נקודות)

מקודדים ספרות בבסיס 10 בייצוג בינארי בו לכל ספרה 0-9 קיימת מילת קוד אחת וייחודית בת 4 סיביות.

נגדיר **קוד עולה ממש**:
יהיו x, y שתי ספרות ויהיו ψ_x, ψ_y הקידודים של הספרות בייצוג בינארי בהתאמה.
אם ורק אם לכל שתי ספרות $x < y$ מתקיים גם ש- $\psi_x < \psi_y$ הקוד נקרא עולה ממש.

הערה: $\psi_x < \psi_y$ משמעותו שהערך המיוצג בבסיס בינארי (ללא סימן) כ- ψ_x **קטן** מהערך המיוצג בבסיס בינארי (ללא סימן) כ- ψ_y.
למשל, עבור $\psi_x = 0111,\ \psi_y = 1000$ מתקיים ש- $\psi_x < \psi_y$ מכיוון ש- $7 < 8$.

מבין הטענות שלהלן, בחרו את הטענה **הנכונה החזקה ביותר**
הערה: טענה א' חזקה יותר מטענה ב' אם"ם נכונות טענה א' גוררת נכונות טענה ב'

א- המרחק של **כל** קוד עולה ממש הוא 1
ב- המרחק של **כל** קוד לקידוד ספרות בבסיס 10 (**לאו דווקא** עולה ממש) הוא 1
ג- **קיים** קוד עולה ממש שהמרחק שלו הוא 2
ד- **קיים** קוד עולה ממש שהמרחק שלו הוא 3
ה- תשובות א'-ד' לא נכונות

הערה: המונח "קוד" בתשובות מתיחס לקוד המייצג ספרות עשרוניות כמתואר בראש השאלה.

פתרון: תשובה ב'
ב-4 סיביות ניתן להרכיב 16 מילים שונות. מתוכן צריך לבחור 10 מילים כדי לקודד את הספרות 0-9. לכל קבוצה של 10 מילים בהכרח קיימות שתי מילים עוקבות (המרחק שלהן הוא 1), ללא קשר אם הקוד הוא עולה ממש או לא.

שאלה 5 (5 נקודות)

בתהליך הצמצום של טבלת המצבים של מערכת עקיבה סינכרונית שיש לה כניסה אחת ויציאה אחת, התקבלה השורה הבאה:

$$\mathrm{P}_3 = (ABC)(DEFG)(HIJ)$$

מבין התשובות הבאות, בחרו את התשובה הנכונה החזקה ביותר:

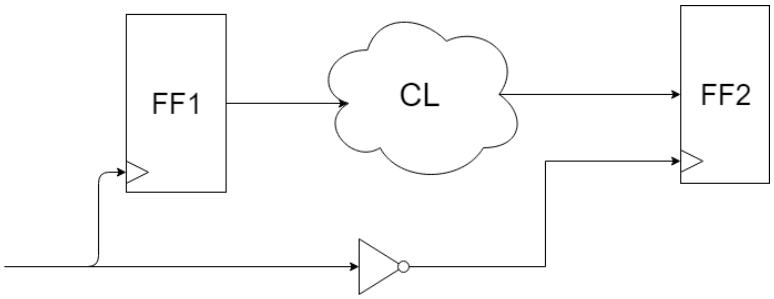
א- המערכת היא **בהכרח** מסוג Mealy
ב- המערכת היא **בהכרח** מסוג Moore
ג- אם המערכת היא מסוג Mealy אז **בהכרח** מתקיים ש- $P_1 = P_2$
ד- אם המערכת היא מסוג Mealy אז **בהכרח** מתקיים ש- $P_2 = P_3$
ה- יש יותר מתשובה אחת נכונה בין התשובות א' עד ד'

הערה: בשאלה זו, תשובה ה' תחשב חזקה יותר מהאחרות.

פתרון: תשובה ד'
לא ניתן לדעת מהו סוג המערכת מהנתונים בשאלה. אפשר למצוא דוגמה לשתי מערכות אחת מילי והשנייה מור ובשתיהן לקבל את P_3 הנתון. ולכן תשובות א' ו- ב' לא נכונות.
תשובה ג' לא נכונה, אפשר למצוא דוגמה למכונה שבה $P_1 \neq P_2$ (למשל ב- P_1 יש שתי מחלקות שקילות בלבד)
תשובה ד' נכונה. בכל שלב K כך ש- $\mathrm{P}_{K-1} \neq P_K$ יש לפחות $\mathrm{K}+1$ מחלקות. אם נניח בשלילה ש- $\mathrm{P}_2 \neq P_3$ זה אומר ש- $|\mathrm{P}_3| \geq 4$ אבל מהנתון רואים שב- P_3 יש רק 3 מחלקות שקילות ולכן אלגוריתם הצמצום הסתיים כבר ב-P_2 כלומר $\mathrm{P}_2 = P_3$.

שאלה 6 (5 נקודות)

מהנדס חומרה תכנן את המעגל הבא:



שימו לב, שבכניסת השעון של $FF2$ מחובר מהפך.

נתונים עבור שני הדלגלגים:

$t_{pCQ} = 10ns$

$t_{cCQ} = 3ns$

$t_{setup} = 2ns$

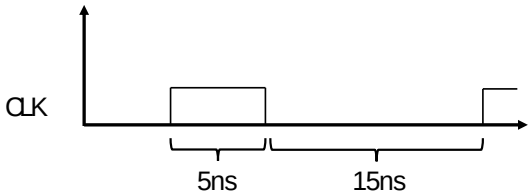
$t_{hold} = 3ns$

נתונים עבור המהפך:

$t_{cd} = 0ns$

$t_{pd} = 0ns$

נתון שזמן המחזור של השעון הינו: $T = 20ns$, ושערך אות השעון שווה ל- $'1'$ במשך $5ns$, ושווה ל- $'0'$, במשך $15ns$. כלומר:



מבין התשובות הבאות, מצאו את התנאים **<u>המתירנים ביותר</u>** על $t_{pd}(CL)$, ו- $t_{cd}(CL)$ שמאפשרים עבודה תקינה של המעגל:

א- $5\ ns \le t_{cd}(CL) \le t_{pd}(CL) \le 8\ ns$

ב- $5\ ns \le t_{cd}(CL) \le t_{pd}(CL) \le 13\ ns$

ג- $0\ ns \le t_{cd}(CL) \le t_{pd}(CL) \le 8\ ns$

ד- $0\ ns \le t_{cd}(CL) \le t_{pd}(CL) \le 13\ ns$

ה- המעגל אינו עומד בתנאי Hold בכל מקרה, ולכן לא קיימים תנאים שיאפשרו את עבודתו התקינה של המעגל.

פתרון:

למעשה, ניתן להגיד שהפליפ-פלופ השני דוגם תמיד $5ns$ מאוחר יותר מאשר הפליפ-פלופ הראשון. לכן בעייה זו היא כמו בעיית Skew רגילה, עם $Skew = 5ns$:

$$t_{pCQ} + t_{pd}(CL) + t_{setup} \leq T + t_{skew}$$

$$10ns + t_{pd}(CL) + 2ns \leq 25ns$$

$$t_{pd}(CL) \leq 13ns$$

עבור תנאי Hold:

$$t_{hold} + t_{skew} \leq t_{cCQ} + t_{cd}(CL)$$

$$3ns + 5ns \leq 3ns + t_{cd}(CL)$$

$$t_{cd}(CL) \geq 5ns$$

שאלות 7-8

נתונה התכנית הבאה:

```
        addi t0, x0, 0
        addi t1, x0, 5
        addi s1, x0, 0x200
Loop:   beq t1, t0, finish
        addi t0, t0, 1
        lw  t2, 0(s1)
        xor t2, t2, -1
        addi s1, s1, 4
        beq x0, x0, Loop
Finish: NOP
        NOP
```

הקוד רץ על מעבד מסוג Pipe-Line RISC-V.

נתונים:

- **קיים** Forwarding מלא, אך **לא קיים** Forwarding בתוך ה- Register File (כלומר – לא קיים Forwarding בין שלב ה- WB לשלב ה- Decode).
- **לא קיימת** יחידת Hazard Detection Unit.
- המעבד מניח תמיד שפקודות- Branch **לא נלקחות**, **ומכיל** מנגנון Flushing במקרה שכן. ההחלטה הסופית על קפיצות מתבצעת בתום שלב ה- EXE.

שאלה 7 (5 נקודות)

מה המספר המינימלי של פקודות NOP שתיאלצו להוסיף לקוד לעיל על מנת שירוץ כשורה?

הערה: לא ניתן לשנות את הסדר של הפקודות

א- 0

ב- 1

ג- 2

ד- 3

ה- 4

נדרשות 2 פקודות NOP מיד לאחר הפקודה השלישית, ו-NOP מיד לאחר פקודת ה-LW.

שאלה 8 (5 נקודות)

עבור שאלה זו בלבד, כל מנגנוני ה- Forwarding וה- Stall פעילים.

כדי לשפר את ביצועי הקוד, הוחלט להוסיף מנגנון Branch Prediction שמנסה לחזות האם ה- Branch הולך לקפוץ או לא:

- אם המנגנון אומר שהולכת להתבצע קפיצה, הפקודות הבאות שתיטענה הן הפקודות החל מיעד הקפיצה (ניתן להניח שהמנגנון יודע מה כתובת הפקודה אליה תתבצע הקפיצה).
- אם המנגנון אומר שלא הולכת להתבצע קפיצה, הפקודות הבאות שתיטענה הן הפקודות העוקבות ל- Branch.
- אם המנגנון טעה, יש צורך בביצוע Flushing.

בהינתן שהמנגנון צודק בחלק- a מהמקרים (כאשר: $0 \le a \le 1$), מה התנאי **המתירני ביותר** על a כך שכדאי יהיה להשתמש במנגנון החדש, עבור הקוד הנתון?

א- $a \ge \frac{5}{11}$

ב- $a \ge \frac{7}{11}$

ג- $a \ge \frac{9}{11}$

ד- בכל מקרה כדאי להשתמש במנגנון החדש.

ה- בכל מקרה לא כדאי להשתמש במנגנון החדש.

פתרון: תשובה א'

בקוד המקורי ישנן שתי פקודות Branch. העליונה מבצעת קפיצה פעם אחת מתוך 6 הרצות של הפקודה, והפקודה התחתונה מבצעת קפיצה 5 פעמים.

סה"כ פקודות ה- Branch נלקחות 6 פעמים, מול 5 פעמים שאינן נלקחות. כלומר, אם המנגנון החדש צודק ב- $a \ge \frac{5}{11}$ מהמקרים, כדאי להשתמש בו.

שאלות 9 (5 נקודות)

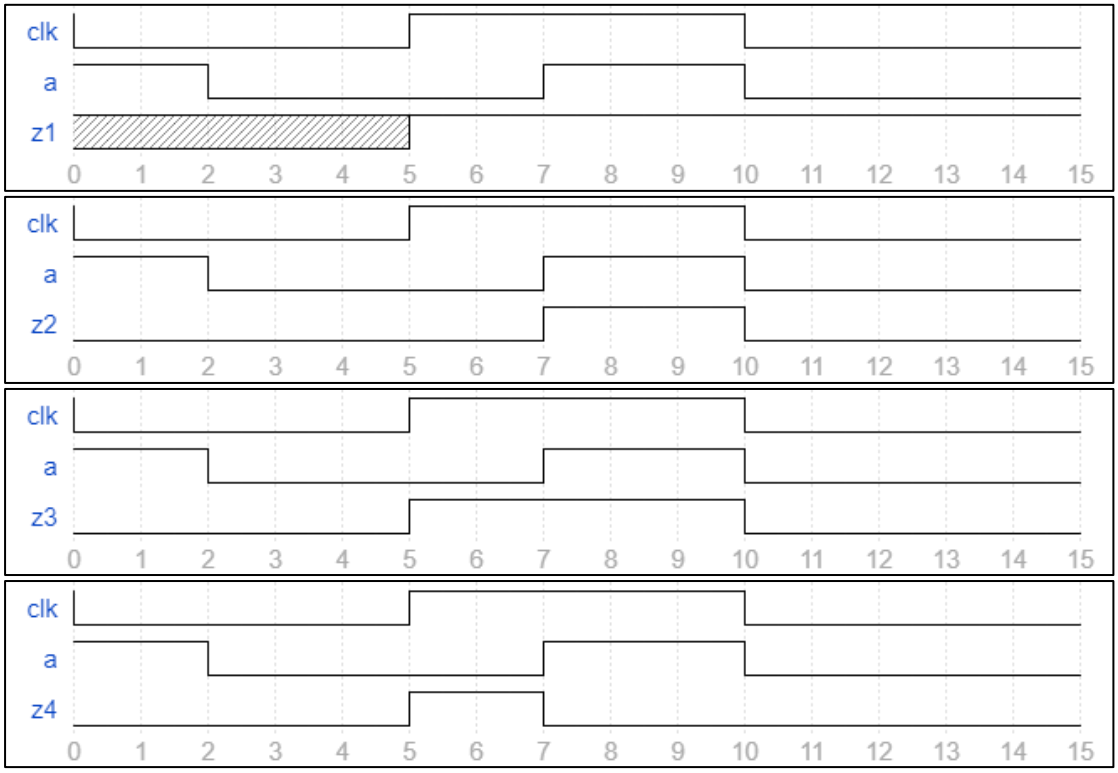
נתון קוד ה-Verilog הבא:

```
module test(clk, a, z);
   input clk;
   input a;
   output reg z;

   always @(clk) begin
      if (clk == 1'b1) begin
         z = ~a;
      end
      else begin
         z = 1'b0;
      end
   end

endmodule
```

נתונות ארבע דיאגרמות אפשריות עבור הפלט z. שימו לב כי בכל הדיאגרמות הקלטים a ו-clk זהים, ולפני t=0 הסיגנל clk היה '1' והסיגנל a היה '1':



הערה: הסימון המפוספס בדיאגרמה z1 מסמן את הערך 'x'.

בחרו את הטענה הנכונה:

א- הפלט z ייראה כמו דיאגרמה z1 בסינתזה ובסימולציה
ב- הפלט z ייראה כמו דיאגרמה z1 בסינתזה וכמו z2 בסימולציה
ג- הפלט z ייראה כמו דיאגרמה z2 בסינתזה וכמו z1 בסימולציה
ד- הפלט z ייראה כמו דיאגרמה z4 בסינתזה וכמו z3 בסימולציה
ה- תשובות א'-ד' לא נכונות

פתרון תשובה ד'

הסבר:
ראשית נשים לב שלמרות שרשימת הרגישויות של משפט ה-always מכילה סיגנל clk, הוא בעצם **אס**ינכרוני (המילה posedge לא מופיעה ברשימת הרגישויות). לכן בסימולציה ה-process ירוץ בכל פעם שחל שינוי בסיגנל clk ולא רק בעלייתו.
ההבדל בין סימולציה לסינתזה הוא בהתייחסות לרשימת הרגישויות. בעוד שבסימולציה אנו מתייחסים לרשימת הרגישויות של process אסינכרוני, בסינתזה מתעלמים מרשימת הרגישויות לחלוטין ומשפט ה-always מתנהג כמו always(*) (כלומר רשימת רגישויות מלאה שמכילה גם את clk וגם את a).
בסינתזה, בכל פעם שיש שינוי ב-clk או ב-a תתעדכן היציאה z. ב-t=0 חל שינוי ב-clk ולכן ה-process יורץ. נשים לב שכל עוד הסיגנל clk ב-0 היציאה גם היא תהיה 0, לכן z1 לא יכולה להיות התשובה. ב-t=5 השעון עולה ואז היציאה צריכה להיות שווה להיפוך של a, כלומר 1, ולכן דיאגרמה z2 נפסלת נשארנו עם שתי תשובות אפשריות: z3 ו-z4. נשים לב שב-t=7 סיגנל היציאה מתעדכן בהתאם לשינוי של a משום שמדובר ברשימת רגישויות מלאה, ולכן התשובה הנכונה היא z4.
בסימולציה יש משמעות לרשימת הרגישויות החלקית ולכן רק כאשר יש שינוי ב-clk יורץ משפט ה-always. בתחילת הסימולציה, נצפה שכל עוד סיגנל ה-clk נמוך גם היציאה תהיה נמוכה ושוב תשובה z1 נפסלת. כאשר השעון עולה, היציאה מתעדכנת להיות ההיפוך של סיגנל a, והתשובות הרלוונטיות הן שוב z3 ו-z4. בניגוד לקודם, כאשר a בלבד משתנה (ו-clk נשאר קבוע), ה-process לא רץ. לכן בזמנים t=7-10 היציאה נשארת על 1 במקום לרדת ל-0 בהתאם לשינוי בערך של a, והתשובה הנכונה היא z3.

שאלה 10 (5 נקודות)

לאחר ייצור מעבד *RISC-V Multi Cycle*, התגלתה בעיה בתפקוד הנובעת מתקלה בייצור:
כאשר הפקודה היא מסוג *lw*, בכל פעם שמגיעים לשלב ה*EXE*, מדלגים מיד אחריו לשלב ה*WB* של הפקודה במקום לבצע קודם את שלב ה*MEM*.
נתון שעבור פקודת *lw* אין תקלות נוספות.
עם זאת, **ייתכנו** תקלות אחרות עבור שאר הפקודות.

מה מהאפשרויות הבאות יכולה להיות מקור הבעיה:

א- הביטים של ה- *AddrCtr* נכנסים הפוך לתוך הבורר שקובע את המצב החדש.
ב- ה- *adder* שמחובר לכניסה 3 של הבורר שקובע את המצב החדש תמיד מוסיף 2 במקום להוסיף 1.
ג- כניסת המידע המתאימה לערך הבקרה 0, בבורר שקובע את המצב החדש, תקועה על הערך '4'.
ד- טעות בשתי שורות בטבלת ה *Dispatch rom 2*.
ה- תשובות א'-ד' לא נכונות

פתרון: תשובה ד'
המעבר בין מצב 2 למצב 3 מתבצע ע"י Dispatch rom 2.

שאלה 11 (5 נקודות)

תכננו מכונת מצבים מסוג MOORE אשר מקבלת כניסה של ביט בודד ומוציאה ביט בודד. ביט המוצא יהיה '1' אם רצף 4 הביטים האחרונים שהתקבלו (שמשפיעים על היציאה) הוא '1001' או '1110'. אחרת, ביט המוצא יהיה '0'. שימו לב שהרצפים מתקבלים משמאל לימין, כלומר, שברצף '1110' הספרה הראשונה שהתקבלה היא '1'.
כמו כן, מכונת המצבים **לא** חוזרת למצב ההתחלתי לאחר שביט המוצא הוא '1'. לדוגמא: אם הרצף שהתקבל הוא '111001' המוצא יהיה '1' גם לאחר קבלת הקלט הרביעי וגם לאחר קבלת הקלט השישי (האחרון).

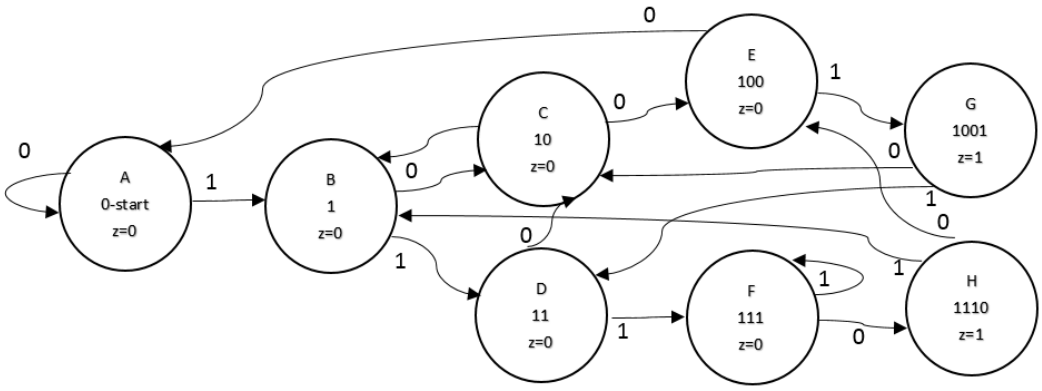
Input	1	1	1	0	0	1	
Output	X	0	0	0	1	0	1

הניחו שלפני קבלת הקלט הראשון המכונה מקבלת הרבה אפסים ('000...000').

מהו מספר המצבים במכונה המצומצמת?

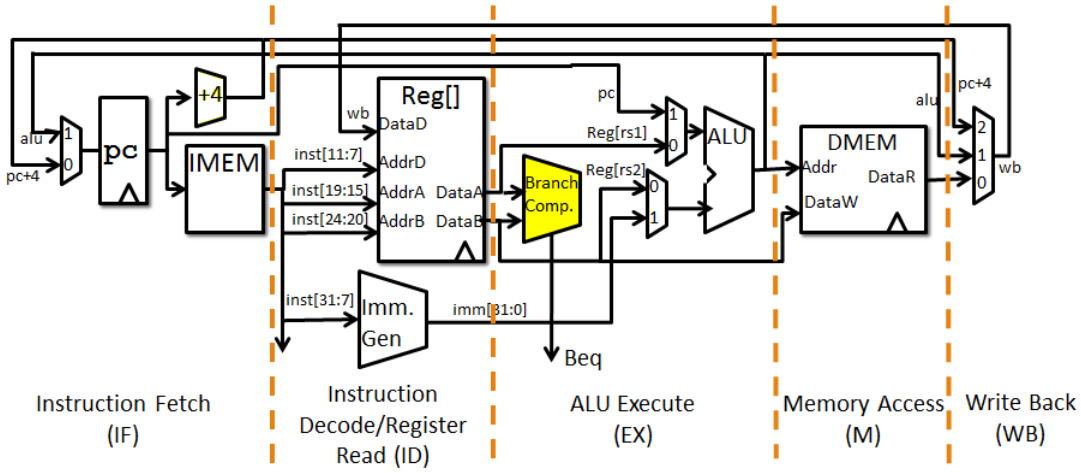
א- 6
ב- 7
ג- 8
ד- 9
ה- לא ניתן לממש את המערכת כמכונת מצבים סופית

פתרון: תשובה ג'



שאלה 12 (20 נקודות)

במעבד *Single Cycle RISC-V* הוחלט לתמוך בפקודות הסתעפות מסוג *beq* ו-*bne* בלבד.



שימו לב שבמעבד הנתון ל- $Branch\ Comparator$ יש יציאה אחת בלבד Beq.

המהנדס שבנה את המעבד, השתמש בטעות ברכיב $Branch\ Comp.$ לא יעיל בעל זמן השהיה גדול יחסית.
נתונים זמני ההשהיה הבאים של השלבים השונים במעבד:

IF	ID	EX	M	WB
2 ns	1 ns	α	2 ns	1 ns

α לא נתון, אבל נתונים לכם זמני ההשהיה של הרכיבים השונים:

$t_{pd}(Branch\ Comparator) = 5\ ns$
$t_{pd}(ALU) = 1\ ns$
$t_{pd}(MUX\ 2 \rightarrow 1) = 0\ ns$
$t_{pd}(Controller) = 0\ ns$

א- (3 נקודות) מהו זמן המחזור המינימלי של המעבד? הסבר.

נחלק את סט הפקודות לשתי קבוצות:

1) פקודות הסתעפות

$$T_{min} = T_{IF} + T_{ID} + \max\{T_{Branch\ Comp.}, T_{mux} + T_{ALU}\} = 8\ ns$$

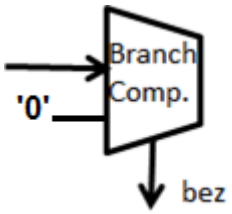
2) כל השאר

$$T_{min} = T_{IF} + T_{ID} + T_{mux} + T_{ALU} + T_{MEM} + T_{WB} = 7\ ns$$

ולכן זמן המחזור המינימלי הוא $8\ ns$

הוצע לבנות **מעבד חדש** שבו מחליפים את ה- $Branch\ Comp.$ ברכיב חדש שיש לו 2 כניסות ויציאה אחת (אחת הכניסות תמיד מחוברת ל- '0').

$$bez(A) = \begin{cases} 1, & A = 0 \\ 0, & otherwise \end{cases}$$



זמן ההשהיה של הרכיב החדש הוא $0.8\ ns$.

כתוצאה מהשינוי, הוחלט לעדכן את סט הפקודות. הפורמט המקורי של פקודות ההסתעפות (beq ו- bne) יהפוך **לפסאודו-פקודה**, לדוגמא:

$beq\ rs1, rs2, label$

יתורגם ל-

$sub\ rs1, rs1, rs2$
$bez\ rs1, label$

כנ"ל גם לגבי bne בהתאמה (במקום bez משתמשים ב- bnz).

ב- (3 נקודות) מהו זמן המחזור המינימלי של המעבד **החדש**? הסבר.

נחלק את סט הפקודות לשתי קבוצות:

1) פקודות הסתעפות

$$T_{min} = T_{IF} + T_{ID} + \max\{T_{Branch\ Comp.}, T_{mux} + T_{ALU}\} = 4\ ns$$

2) כל השאר

$$T_{min} = T_{IF} + T_{ID} + T_{mux} + T_{ALU} + T_{MEM} + T_{WB} = 7\ ns$$

ולכן זמן המחזור המינימלי הוא $7\ ns$

בסעיפים הבאים, הניחו כי זמן המחזור **לפני השינוי הוא $10\ ns$** וזמן המחזור **אחרי השינוי הוא $7\ ns$** (ללא תלות בתשובה לסעיפים הקודמים).

נסמן:
$T_{old}(program)$ - זמן הריצה של תוכנית אסמבלי על המעבד הישן.
$T_{new}(program)$ - זמן הריצה של תוכנית אסמבלי על המעבד החדש.

ג- (3 נקודות) תנו דוגמה לתוכנית $Prog$ כך ש- $T_{new}(Prog) < T_{old}(Prog)$

תוכנית שלא מכילה פקודות הסתעפות
דוגמה אפשרית:

```
add x0, x0, x0
add x1, x1, x1
add x2, x2, x2
```

ד- (3 נקודות) תנו דוגמה לתוכנית $Prog$ כך ש- $T_{new}(Prog) > T_{old}(Prog)$

תוכנית שכולה פקודות הסתעפות
דוגמה אפשרית:

```
Label_1: beq x0, x0, Label_2
Label_2: beq x1, x1, Label_3
Label_3: beq x2, x2, Label_4
Label_4:
```

ה- (4 נקודות) תנו דוגמה לתוכנית שאם מריצים אותה על המעבד החדש מקבלים תוצאה **שונה** מזו שמקבלים אם מריצים אותה על המעבד הישן. הסבירו.

שימו לב שבמעבד החדש אנחנו משנים את התוכן של הרגיסטר הראשון לאחר ביצוע פקודת ההסתעפות. לעומת זאת, במעבד הישן הערכים של הרגיסטרים נשארים ללא שינוי.
כל תוכנית שמשתמשת ברגיסטר הראשון אחרי פקודת ההסתעפות.
דוגמה אפשרית:

$$\begin{array}{rl} & addi\ a0, a0, 7 \\ & addi\ a1, a1, 7 \\ & beq\ a0, a1, label \\ label: & add\ a0, a0, a1 \end{array}$$

לאחר סיום ריצת התוכנית במעבד הישן: $a0 = 14$
לאחר סיום ריצת התוכנית במעבד החדש: $a0 = 7$

ו- (4 נקודות) בסעיף זה בלבד, מניחים שאין את הבעיה מהסעיף הקודם.
בהינתן תוכנית $Prog$ כלשהי, שעבורה ידוע כי מתקבל את אותו הפלט על שני המעבדים. איך מחליטים איפה כדאי להריץ אותה? על המעבד החדש או על המעבד הישן? הסבר.
<u>רמז:</u> תחשבו על כמות הפקודות מהסוגים השונים בתוכנית

מצד אחד, במעבד החדש זמן מחזור השעון קטן יותר. אך מצד שני, מספר רב של פקודות הסתעפות יגדיל את מספר הפקודות המעשי אשר המעבד יצטרך לבצע.

נניח N הוא מספר הפקודות בתוכנית המקורית ו- a הוא מספר פקודות ההסתעפות בתוכנית זו.
מספר הפקודות בתוכנית החדשה שרצה על המעבד החדש הוא $N + a$

$$T_{old}(Prog) = N \cdot T_{min,old} = 10\,N$$
$$T_{new}(Prog) = (N + a) \cdot T_{min,new} = 7 \cdot (N + a)$$

כדי שיהיה כדאי לנו להריץ את התוכנית על המעבד החדש נדרוש ש- $T_{new} \leq T_{old}$

$$7\mathrm{N} + 7a \leq 10N \;\rightarrow\; a \leq \frac{3}{7}N$$

כלומר, אם החלק היחסי של פקודות ההסתעפות מתוך סה"כ הפקודות בתוכנית הוא לכל היותר $\frac{3}{7}$, נעדיף להריץ את התוכנית על המעבד החדש. אחרת, על המעבד הישן.

שאלה 13 (10 נקודות)

שני מהנדסי חשמל מעוניינים לכתוב קוד בשפת אסמבלי.

המהנדס הראשון כותב את הפונקציה SECRET_FUNC, שמקבלת כתובת בזיכרון ומבצעת פעולה סודית על המספר שמאוחסן בכתובת זו. עליכם לעזור למהנדס השני לכתוב את הפונקציה APPLY_ON_ARRAY שמבצעת את הפעולה הסודית על כל איברי מערך.

נתון שהמערך מתחיל בתא 0x20000 ושגודל המערך הינו 0x20 מילים (שניהם בבסיס 16).

א- (5 נקודות) השלימו את הקוד במקומות הריקים **(שורות 2-3, 8, 9, 11)** לפי הנדרש בהערות. שימו לב שבשורות 2-3, צריך לאתחל את כתובת תחילת המערך לרגיסטר s0. ניתן להשתמש בפקודה יחידה, או ב- 2.

```
1  APPLY_ON_ARRAY:  addi sp, sp, -0x20
2                   _____________      # Initialize array address to s0
3                   _____________      # Initialize array address to s0
4                   addi s1, x0, 0x20
5                   add s2, x0, x0
6  LOOP:            beq s2, s1, FINISH
7                   addi s2, s2, 1
8                   sw __, 0(sp)       # Backup registers
9                   ____________       # Prepare arguments for function
10                  jal ra, SECRET_FUNC
11                  lw __, 0(sp)       # Reload backed-up registers
12                  addi s0, s0, 4
13                  jal x0, LOOP
14 FINISH:
```

ב- (5 נקודות) לאחר התיקונים שביצעתם לפונקציה, האם הקוד מקיים קונבנציית הקריאה לפונקציה? הסבר. אם לא, תאר אילו שינויים נדרשים על מנת שהקוד ירוץ לפי הקונבנציה.

לא. על מנת שהפונקציה תרוץ לפי קונבנציית הקריאה לפונקציה, נדרש:
(1) לגבות את s0,s1,s2 בתחילת הריצה, ולהחזיר אותם לערך הראשוני בסוף הריצה.
(2) לשחרר את המחסנית בסוף התוכנית

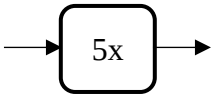
(3) – לא הכרחי- לגבות את a0 לפני הקריאה לפונקציה secret

פתרון א:

```
APPLY_ON_ARRAY:   addi sp, sp, -32
                  lui s0, 0x20          # Initialize s0 to hold the address of the array
                  addi s1, x0, 0x20     # Initialize s1 to hold the size of the array
                  add s2, x0, x0
LOOP:             beq s2, t1, FINISH
                  addi s2, s2, 1
                  sw ra, 0(sp)
                  add a0, x0, s0        # Prepare arguments for the called function
                  jal ra, SECRET_FUNC
                  lw ra, 0(sp)
                  addi s0, s0, 1
                  jal x0, LOOP
FINISH:
```

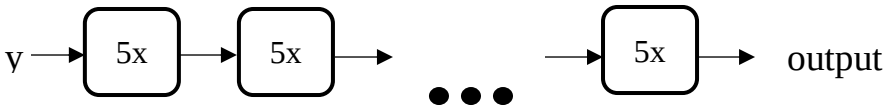
שאלה 14 (15 נקודות)

נתון רכיב צירופי אשר מקבל מספר כלשהו, y, ומוציא ערך שהוא 5y.
נקרא לרכיב "5x" ונסמן אותו באופן הבא:



נתון שהרכיב מקיים $t_{pd} = 4\mu s$
בעזרת שימוש ברכיב הנתון, נרצה לבנות מערכת חדשה שתחשב עבור מספר כלשהו, y, את הערך $5^a y$.

פתרון אחד שהוצע הוא מערכת בעלת *a* רכיבים הבנויה בצורה הבאה:

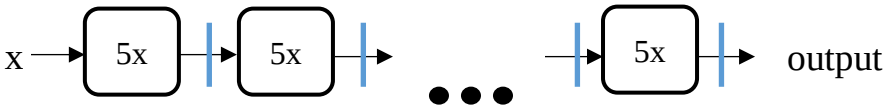


א- (4 נקודות) מהו ה- *Throughput* ומהי ה- *Latency* של המערכת (כתלות ב- *a*)?

$$\text{Latency} = 4a\ \mu s$$
$$\text{Throughput} = \frac{1}{4a}\left(\frac{1}{\mu s}\right)$$

על מנת להגדיל את ה*Throughput*, הוצע לצנר (*pipeline*) את המערכת ע"י הוספת רכיבי פליפ-פלופ אידאליים ($t_{setup} = t_{hold} = 0, t_{pCQ} = 0$).

ב- (4 נקודות) **שרטטו** את המערכת המצונרת **וחשבו** את ה- *Throughput* וה- *Latency* האופטימליים.



$$\text{Latency} = 4a\ \mu s$$
$$\text{Throughput} = \frac{1}{4}\left(\frac{1}{\mu s}\right) = \frac{1}{T}$$

כעת נתון שהפליפ-פלופים לא אידאליים אבל עדיין זהים. לכל פליפ-פלופ יש את הערכים הבאים: $t_{setup} = 10\mu s,\ t_{hold} = 0,\ t_{pCQ} = 15\mu s$.

ג- (3 נקודות) חשבו את זמן המחזור המינימלי T_{min} שעבורו המערכת תעבוד באופן תקין.

$$T \geq t_{pCQ} + t_{pd}(5x) + t_{setup} = 15 + 4 + 10 = 29\ \mu s$$
$$T_{min} = 29\ \mu s$$

ד- (4 נקודות) מהם ערכי *a* עבורם עדיף לעבוד עם המערכת המקורית (מסעיף א') מבחינת *Throughput*? הסבירו.

כעת ה- Throughput של המערכת הוא $Throughput = \frac{1}{29}(\frac{1}{\mu s})$.
לכן, המימוש הראשון יהיה יותר יעיל כאשר:

$$\frac{1}{4a} \geq \frac{1}{29}$$

$$a \leq \frac{29}{4}$$

כלומר כאשר מספר הרכיבים הוא 7 ומטה.