מערכות ספרתיות ומבנה המחשב (044252)

סמסטר חורף תשפ"ב

בחינה סופית – מועד א - פתרון

7 בפברואר 2022

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 ולמחשבון.

הנחיות והוראות:

- הבחינה כתובה על גבי **21** עמודים כולל עמוד זה (לא רלוונטי עבור קובץ הפתרון) (**בדקו בתחילת הבחינה שלא חסרים לכם עמודים**). בסה"כ ישנן 17 שאלות: 14 שאלות אמריקאיות, ו- 3 שאלות פתוחות מרובות סעיפים.
- 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 הפתוחות בגוף המבחן, **במלבנים המסומנים לכך בלבד.**
- 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). **ודאו כי על מחברת הבחינה ועל טופס התשובות האמריקאי מודבקת מדבקת הנבחן שלכם.**
- לא מורדות נקודות (אין "קנס") בגין תשובה שגויה. לכן, בשאלות האמריקאיות כדאי לסמן תשובה כלשהי לכל שאלה.
- ציון השאלות האמריקאיות 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 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, טלפתית, או אחרת. בפרט, אין להחזיק בטלפון סלולארי.

בהצלחה!

שאלה 1 (5 נקודות):

נתון הרכיב my_module הבא:

```
module my_module (
      input  logic       my_in,
      output logic [3:0] my_out
      );

      logic [3:0] my_vec;
      assign      my_vec[0] = my_in;

      genvar i;
      generate
            for (i=0; i<3 ; i++)
            begin
                  unit #(
                        .SHIFT_NUM(i)
                  ) unit_inst (
                        .unit_in(my_vec[i]),
                        .unit_out(my_vec[i+1])
                  );
            end
      endgenerate

      assign my_out = my_vec;

endmodule
```

כאשר הרכיב unit הינו:

```
module unit (
      input  logic unit_in,
      output logic unit_out
      );

      parameter SHIFT_NUM;
      logic [3:0] unit_vec;

      assign unit_vec = 4'b1101 >> SHIFT_NUM;
      xor(unit_out,unit_vec[0],unit_in);

endmodule
```

(תזכורת: במוצא הפעולה: 4'b1000 >> 2. נקבל 4'b0010)

מה המוצא my_out של הרכיב my_module בהרצת סימולציה שבה my_in=1'b0?

א. 4’b0110

ב. 4’b1101

ג. 4’b0010

ד. 4’b1110

ה. ישנה שגיאה בקוד כתוצאה מהתנגשות. כלומר - ביט אחד מקבל שני ערכים שונים במקביל.

פתרון:

תשובה א'.

הביט הימני ביותר של my_vec שווה לכניסה, כפי שנקבע בהשמה, ולכן הוא 0. שאר הביטים של my_vec נקבעים לפי הערכים שמוציא הרכיב unit בכל איטרציה של הלולאה.

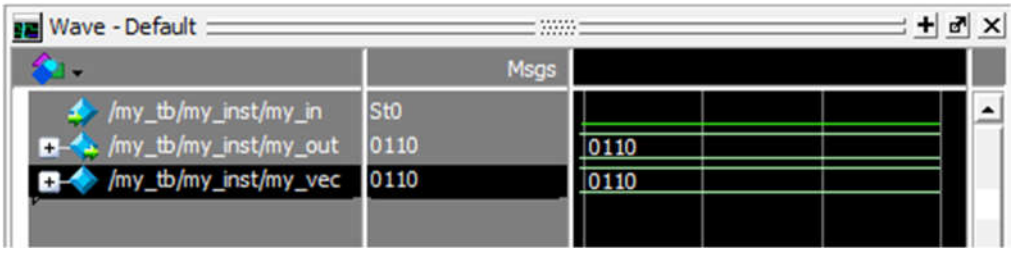
באיטרציה ראשונה: unit מקבל 1'b0 ומוציא 1'b1.

באיטרציה שניה: unit מקבל 1'b1 ומוציא 1'b1.

באיטרציה שלישית: unit מקבל 1'b1 ומוציא 1'b0.

(המודול unit קובע את הביט ה-(i+1) של my_vec להיות תוצאת פעולת XOR של הביט ה-i של my_vec עם הביט ה-i של 4'hd=4'b1101)

והתוצאה היא 4'b0110:



שאלה 2 (5 נקודות):

מהנדס מעוניין לייצר קוד המקבל מספר המורכב מסיביות בינאריות באורך כלשהו, ומסוגל לייצג את **שני** סוגי המספרים הבאים:
סוג א': מספרים בקידוד המשלים ל-2, בטווח $[-2^8, 2^8 - 1]$
סוג ב': מספרים בקידוד unsigned, בטווח $[322, 833]$

מהו המספר **המינימלי** של ביטים הנדרשים עבור קוד זה?

רמז – התייחסו גם ליכולת להבדיל בין הסוגים השונים.

א. 9
ב. 10
ג. 11
ד. 12
ה. 13

פתרון:

נצטרך סיבית אחת עבור בחירת הסוג.

עבור סוג א' $[-2^8, 2^8 - 1]$ נצטרך 9 סיביות.

עבור סוג ב' $[322, 833] = 322 + [0, 2^9 - 1]$ ולכן נצטרך 9 סיביות.

ולכן התשובה היא ב', 10 סיביות.

<u>שאלה 3 (5 נקודות):</u>

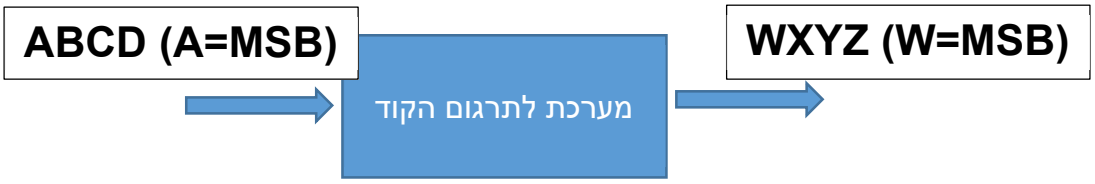
תכננו מערכת שמתרגמת בין שני ייצוגים שונים.
המערכת מקבלת כקלט <u>ספרה עשרונית</u> המיוצגת בעזרת 4 סיביות שכתובות בייצוג משוקלל (6, 4, 2, -3) כאשר 6 מייצג את משקל סיבית ה-MSB.

תזכורת: ספרה המיוצגת בייצוג משוקלל היא: $N = \sum_i w_i d_i$,

כאשר N מייצג ספרה בין 0 ל- 9. שימו לב, ייתכן שלחלק מהקלטים קיים ייצוג כפול.

המערכת מוציאה כפלט את אותה ספרה עשרונית אך מיוצגת בעזרת קוד BCD.

משתני הכניסה הם A,B,C,D (A=MSB)
משתני היציאה הם W,X,Y,Z (W=MSB)



<u>דוגמה:</u>

ABCD=0110 מייצג את הספרה העשרונית "6"
ולכן נקבל במוצא: WXYZ=0110. שימו לב שגם הקלט ABCD=1000 הינו חוקי, ובמוצא המערכת נקבל גם WXYZ=0110.

מבין האפשריות הבאות, מהו הביטוי המצומצם ביותר כסכום של מכפלות עבור המוצאים W ו-X של המערכת?

א. W= AD + BCD
X= B'CD + BD' + ABC'

ב. W= AD' + BCD'
X= B'CD + BD' + ABC'

ג. W=ABC+ACD'
X=B'CD+ABC'+AC'D'+BD'

ד. W= AD' + BCD
X= B'CD + BD' + ABC'

ה. תשובות א-ד אינן מבטאות נכונה את התרגום הנדרש לקביעת W ו-X.

פתרון:

תשובה ג'.

ABCD 6,4,2,-3	WXYX 8,4,2,1
0000	0000
0001	dddd
0010	0010
0011	dddd
0100	0100
0101	0001
0110	0110
0111	0011
1000	0110
1001	0011
1010	1000
1011	0101
1100	dddd
1101	0111
1110	dddd
1111	1001

נבנה מפת קרנו מתאימה.

עבור W:

CD \ AB	00	01	11	10
00	0	0	d	0
01	d	0	0	0
11	d	0	1	0
10	0	0	d	1

W= ABC+ACD’

עבור X:

CD \ AB	00	01	11	10
00	0	1	d	1
01	d	0	1	0
11	d	0	0	1
10	0	1	d	0

X = B’CD + ABC’ + AC’D’ + BD’

לכן התשובה הנכונה היא ג'.

<u>שאלה 4 (5 נקודות):</u>

סטודנטית בקורס מעוניינת לממש את הפונקציה:

$$f(A,B,C,D,E) = \Sigma(19,21,22,27,29,30)$$

הסטודנטית מעוניינת לממש את הפונקציה הנתונה, בעזרת מפענחים ושערים לוגים. כלל המפענחים בשאלה בעלי כניסת Enable.

לרשות הסטודנטית עומד שער OR בעל 32 כניסות **שאינו** יספר במניין השערים הלוגים בתשובות. הסטודנטית יכולה להשתמש בקבועים '0' ו- '1' ללא הגבלה.

סמנו את התשובה **המאפשרת מימוש של הפונקציה** ומכילה רכיבים לפי סדר עדיפות הסטודנטית, כאשר התשובות מסודרות לפי עדיפות. כלומר, תשובה א' היא הכי פחות עדיפה ותשובה ה' הכי עדיפה.

א. ניתן לממש את הפונקציה בעזרת מפענח יחיד מסוג $5 \rightarrow 32$, <u>וללא</u> תוספת שערים לוגים.
ב. ניתן לממש את הפונקציה בעזרת מפענח יחיד מסוג $4 \rightarrow 16$, <u>וללא</u> תוספת שערים לוגים.
ג. ניתן לממש את הפונקציה בעזרת **שני** מפענחים מסוג $3 \rightarrow 8$, <u>ובתוספת</u> שערים לוגים.
ד. ניתן לממש את הפונקציה בעזרת מפענח יחיד מסוג $3 \rightarrow 8$, <u>ובתוספת</u> שערים לוגים.
ה. ניתן לממש את הפונקציה בעזרת מפענח יחיד מסוג $3 \rightarrow 8$, <u>וללא</u> תוספת שערים לוגים.

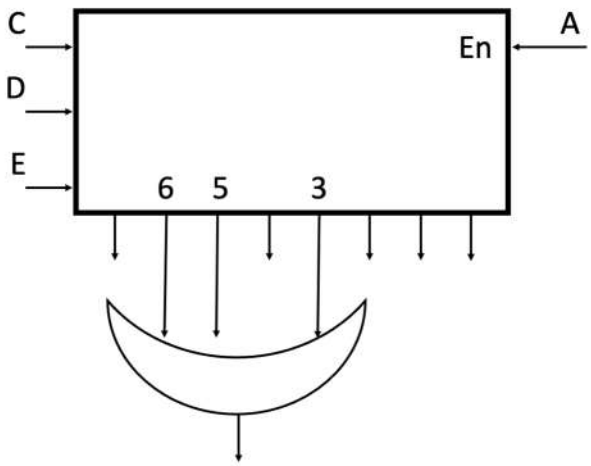
פתרון:

תשובה ה'.

הפונקציה מוציאה 1 עבור הקלטים הבאים:

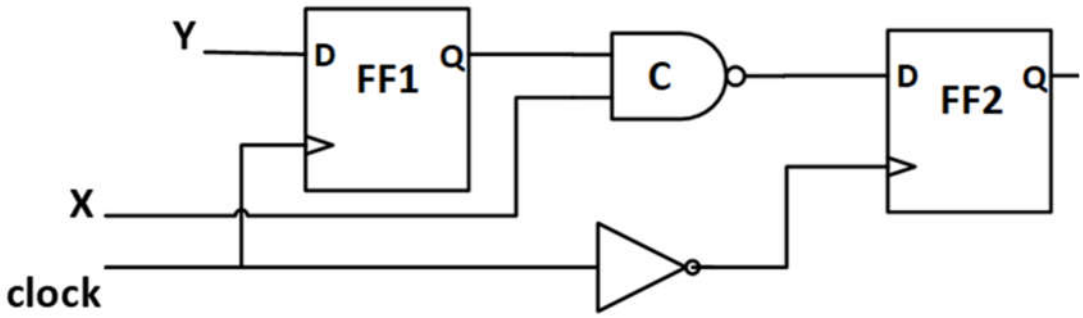
	ABCDE	f
19	10011	1
21	10101	1
22	10110	1
27	11011	1
29	11101	1
30	11110	1

כלומר, אם A=0 הפונקציה בוודאות מוציאה 0. בנוסף, כאשר A=1, הפונקציה אינה תלויה בסיבית B, אלא רק בסיביות C D ו- E. לכן אנחנו זקוקים למפענח יחיד מסוג 3->8, ללא תוספת שערים לוגים.

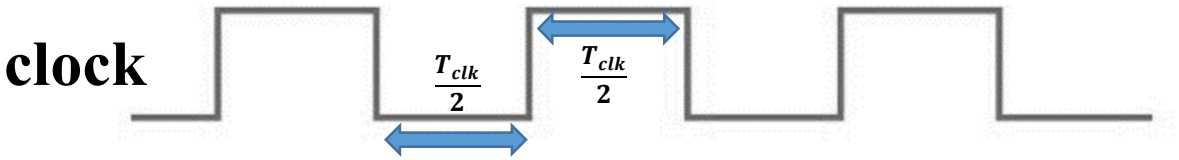


שאלה 5 (5 נקודות):

נתון המעגל הבא:



-	t_{cd}	t_{pd}/t_{pCQ}	t_{setup}	t_{hold}
$FF1/FF2$	5	40	20	5
$NAND$	2	22		
NOT	2	3		



השעון שמחובר ל-FF1 הוא בעל Duty cycle=0.5, כלומר חצי מהמחזור הוא שווה ל- '1', וחצי מהזמן שווה ל- '0'.
שימו לב שהשעון נכנס למהפך לפני שהוא עובר ל-FF2.

בנוסף, יש להתעלם מבדיקת תנאי hold במעגל, ונתון שהכניסות X ו- Y עומדות בתנאי התזמון של המעגל.

מהו זמן המחזור המינימלי המאפשר עבודה תקינה של המערכת?

א. 62

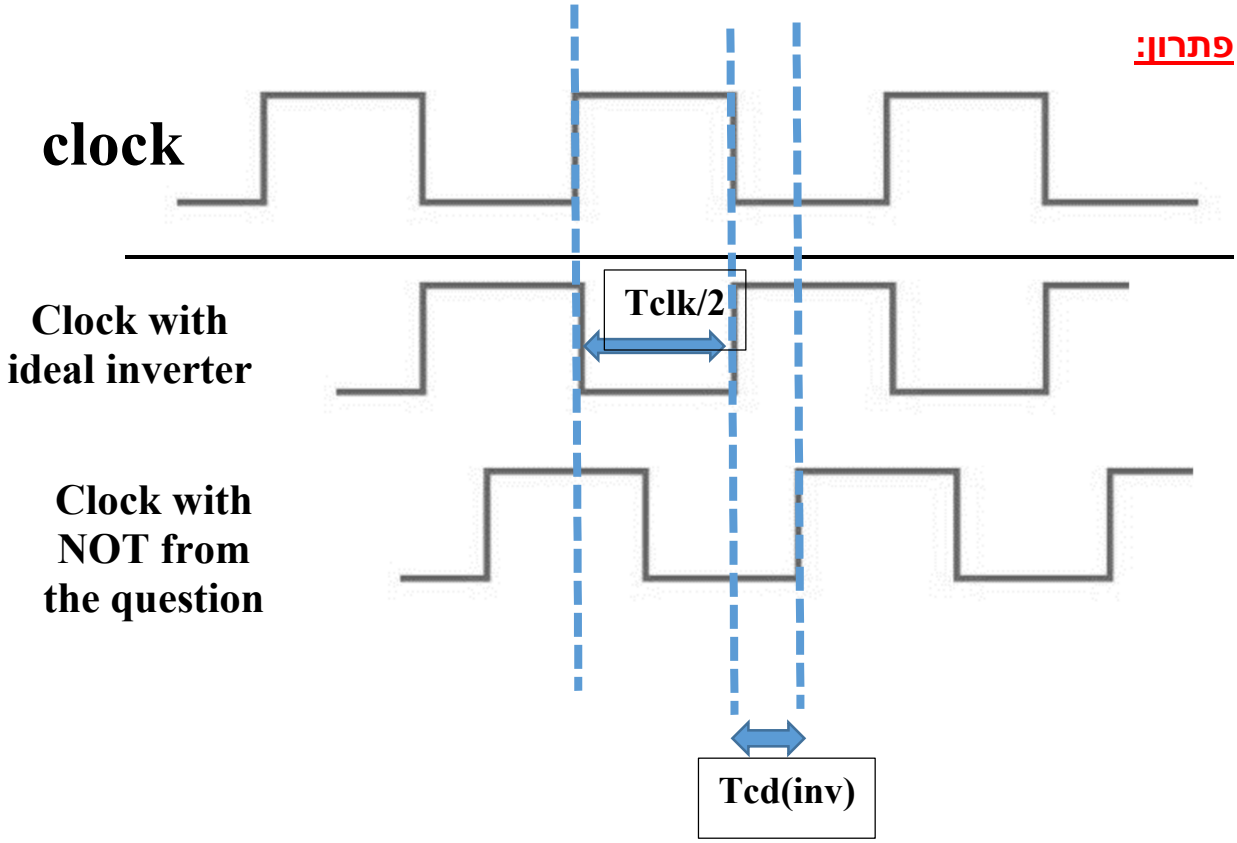
ב. 80

ג. 102

ד. 124

ה. 160

פתרון:



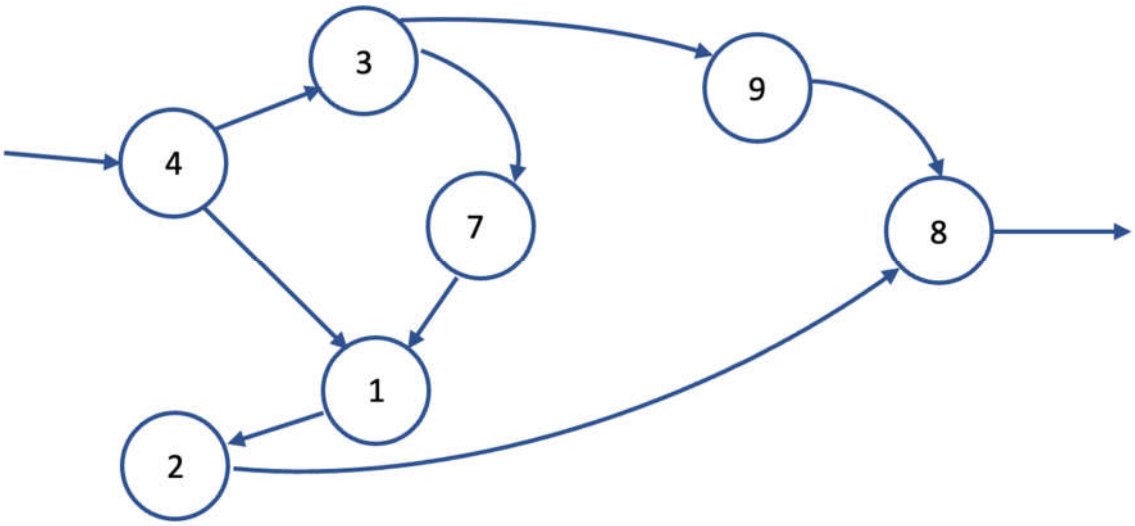
$$Tpd(FF) + tpd(NAND) + Tsu(FF) \leq Tclk/2 + tcd(inv)$$

$$40 + 22 + 20 \leq Tclk/2 + 2$$

$$Tclk = 2 \cdot 80 = 160$$

שאלה 6 (5 נקודות):

נתונה המערכת הצירופית הבאה:



זמן ההשהיה של כל רכיב צירופי מצוין על גביו.

מהי כמות הרגיסטרים המינימלית אשר דרושה לצורך צינור המערכת כדי שתתקבל תפוקה (throughput) מקסימלית בעדיפות ראשונה, והשהייה מינימלית בעדיפות שניה?

לצורך הצינור, כל הרגיסטרים הנתונים אידיאליים (בעלי זמן השהייה 0, ותנאי hold מתקיים בכולם).

א. 4

ב. 6

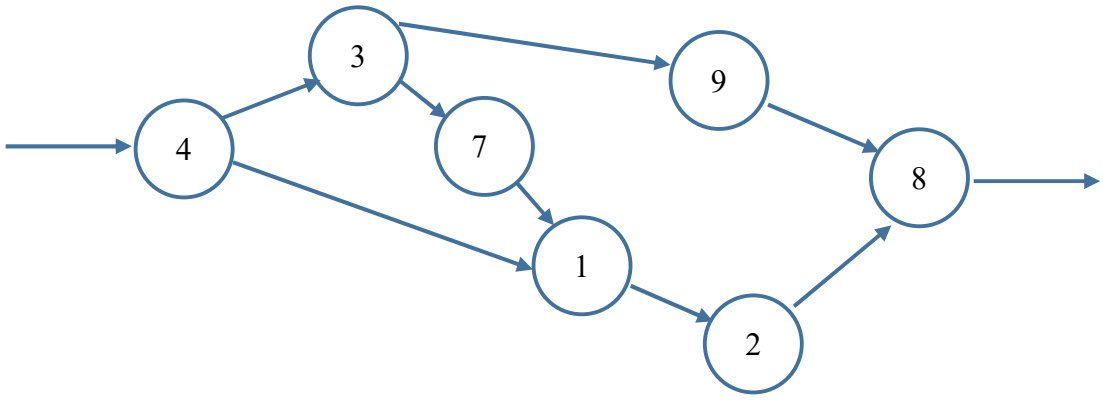
ג. 7

ד. 8

ה. 9

פתרון:

תחילה נשרטט מחדש את המערכת כך שכל החיצים יהיו לאותו כיוון. לאחר מכן נשתמש בשיטת הקווים שלמדנו. נשים לב כי היחידה האיטית ביותר צורכת תשע יחידות זמן. במידה ונבודד אותה בעזרת רגיסטרים נוכל לקבל את התפוקה המקסימלית שתהיה שווה ל-$\frac{1}{9}$:



נקפיד על הוספת רגיסטר במוצא המעגל. מספר הרגיסטרים הדרוש לצורך הצינור שווה למספר החיתוכים של הקווים עם החיצים, כלומר 8 רגיסטרים.

תשובה ד'.

שאלה 7 (5 נקודות):

הוחלט לשנות את פרוטוקול התקשורת UART שנלמד בכיתה (start bit = 0, 8 bits of data, stop bit = 1) באופן הבא:

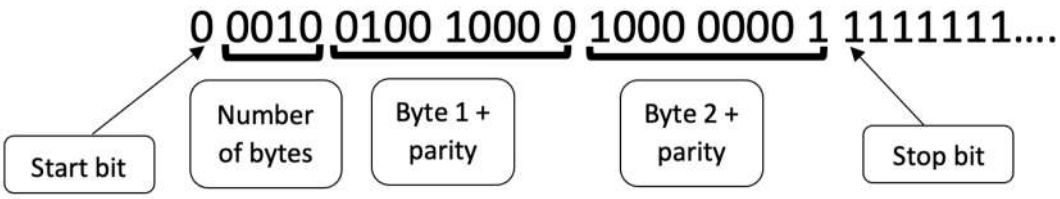
משדרים start bit = 0, ולאחר מכן 4 סיביות המתארות את מספר **הבתים** (כל בית מכיל 8 סיביות) שנשדר ברצף. לאחר מכן, על מנת לשפר את יכולת גילוי השגיאות של הפרוטוקול, משדרים כל בית עם סיבית הזוגיות שלו. לבסוף משדרים stop bit = 1.

שימו לב: משמעות שידור בתים ברצף היא שאין ביניהם start bit/stop bit.

בנוסף מוגדר בפרוטוקול שמספר הבתים המשודר גדול מ- 0 (כלומר הסיביות 1:4 לא יכולים להיות כולם 0).

השידור מתחיל מה-LSB.

לדוגמה, אם נרצה לשדר את הבתים 0x12 ו- 0x01 (בסדר הזה) נשדר את סדרת הסיביות הבאה:



מספר הבתים המשודר הוא $0 < N \leq 15$.

סטודנט מעוניין לשלוח הודעה בעלת N בתים בזמן הקצר ביותר (זמן שליחת סיבית אחת בפרוטוקול החדש ובפרוטוקול הסטנדרטי זהה). מה ה- N המינימלי עבורו הסטודנט יעדיף להשתמש בפרוטוקול החדש?

א. לכל $0 < N \leq 15$ עדיף להשתמש בפרוטוקול החדש.
ב. $N = 3$
ג. $N = 7$
ד. $N = 8$
ה. לכל $0 < N \leq 15$ עדיף להשתמש בפרוטוקול הסטנדרטי.

פתרון:

תשובה ג'

עבור הפרוטוקול הנלמד בכיתה, הזמן הדרוש לשליחה N בתים הוא:

$$t_{old} = 10N$$

עבור הפרוטוקול החדש, הזמן הדרוש לשליחה N בתים הוא:

$$t_{new} = 2 + 4 + 9N = 6 + 9N$$

אם כן, השיפור הוא:

$$\frac{t_{old}}{t_{new}} = \frac{10N}{6 + 9N}$$

נדרש שיפור גדול ממש מ – 1:

$$\frac{t_{old}}{t_{new}} > 1 \Rightarrow \frac{10N}{6 + 9N} > 1 \Rightarrow N > 6$$

ולכן נדרש לשלוח יותר מ 6 בתים כדי לשפר ביצועים עם הפרוטוקול החדש.

שאלה 8 (5 נקודות):

עליכן לתכנן מכונת מצבים מסוג מילי, אשר תזהה רצפים באורך זוגי גדול מ-0 של 1-ים. **עם סיומו** של כל רצף זוגי של '1' (כלומר, עם קבלת '0' לאחר רצף של '1') המכונה תוציא '1'.

להלן דוגמה של ההתנהגות הרצויה של המכונה:

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

כמה מצבים נדרשים לצורך מימוש מכונה זו?

א. 2

ב. 3

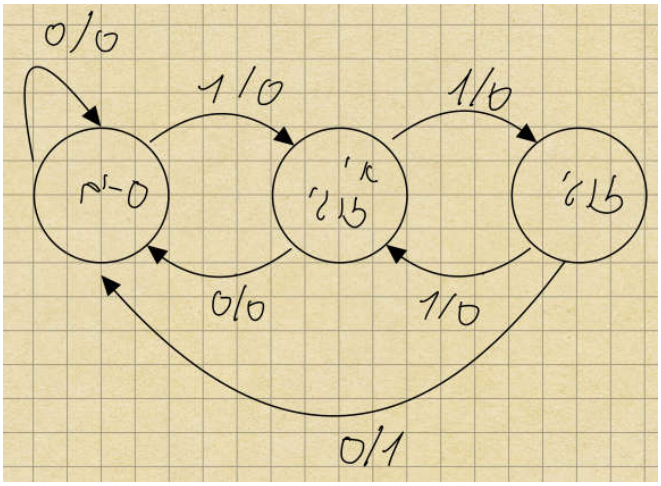
ג. 4

ד. 5

ה. 6

פתרון:

תשובה ב'.



שאלה 9 (5 נקודות):

נתון קטע הקוד הבא:

```
addi    t0,    x0,    0x8
slli    t0,    t0,    3
add     t0,    t0,    t0
srli    t0,    t0,    2
```

מה ערכו של רגיסטר t0 עם סיום התוכנית?

שימו לב שערכי המספרים בתשובות הינם בבסיס עשרוני.

א. 4
ב. 8
ג. 16
ד. 32
ה. 64

פתרון:

התוכנית מכפילה את המספר 8 ב-16 ואז מחלקת את התוצאה ב-4. בסה"כ מבצעים את התרגיל $8 \cdot 4$.

תשובה ד'.

שאלה 10 (5 נקודות):

נתון מערך A השמור בזיכרון. כל איבר במערך הוא בן 4 בתים.
כתובת הבסיס שלו נמצאת ב-t1, ואיברי המערך הם: A={0,1,2,3,4,5}.

נתון קוד האסמבלי הבא:

```
        Addi t2, x0, 6
        Add s1, x0, x0
Loop: beq t2, x0, end
        Lw t3, 0(t1)
        Add s1, s1, t3
        Addi t1, t1, 8
        Addi t2, t2, -2
        J loop
End:
```

מה יהיה הערך של s1 בסוף הריצה?

א. 0

ב. 1

ג. 6

ד. 5

ה. 10

פתרון:

תשובה ג'.

הקוד מחשב את הסכום של איברי המערך באינדקסים הזוגיים.

```
        Addi t2, x0, 6 // t2=5 number of iterations
        Add s1, x0, x0 //s1=0 initiate sum=0
Loop: Beq t2, x0, end // while t2>0
        Lw t3, 0(t1) // t3=A[t1]
        Add s1, s1, t3 // update the sum
        Addi t1, t1, 8 // update t1 to point to the next next address
        Addi t2, t2, -2 // next iteration
        J loop
End:
```

שאלה 11 (5 נקודות):

עבור הפקודות הבאות, איזו פקודה ניתן לממש כפקודה אמיתית (לא פסאודו פקודה) במעבד ה**single cycle RISC-V** ?

ניתן לבצע שינויים בבקר, הוספת בוררים וחיווטים במסלול הנתונים של המעבד ולהתאים את זמן המחזור, אך **אסור** לבצע שינויים ביחידות: Register file, ImmGen, Memory, ALU.

א. פקודת mv2 rd1, rd2, rs אשר מעתיקה את הערך ברגיסטר rs לרגיסטר rd1 וגם לרגיסטר rd2.

ב. פקודת subi20 rd, rs, imm אשר מבצעת חיסור בין 32 הביטים של רגיסטר rs לבין immediate בגודל 20 ביט (שלאחר מכן עובר דרך ה- ImmGen ומתורגם למספר בן 32 סיביות) ושומרת את התוצאה ב rd.

ג. פקודת cp rs1, rs2 אשר מעתיקה מילה מהזכרון מהכתובת שנתונה ברגיסטר rs1 לכתובת בזיכרון אשר נתונה ברגיסטר rs2.

ד. פקודת addsub rd, rs1, rs2 אשר מבצעת את הפעולה

reg[rd]=reg[rd]+reg[rs1]-reg[rs2] (שומרת את התוצאה לרגיסטר rd).

ה. לא ניתן לממש אף פקודה מהפקודות הנ"ל.

פתרון

א' לא נכון מכיוון שאין אפשרות לעשות 2 כתיבות שונות באותו מחזור

ב' לא נכון מכיוון שאין אפשרות לרשום ערך של 20 ביט ועוד רגיסטרים בפקודה (אין מספיק ביטים)

ג' לא נכון מכיוון שאין אפשרות לגשת לזיכרון פעמיים באותו מחזור

ד' לא נכון מכיוון שאין אפשרות לחשב מספר פעולות ב ALU באותו מחזור

תשובה ה' נכונה.

אף אחת מהפקודות לא ניתנת למימוש.

<u>שאלה 12 (5 נקודות):</u>

נתונה תכנית אשר מורכבת מ-N פקודות.

בנוסף נתון פילוח פקודות התוכנית אשר בפועל רצות על המעבד לפי סוג:

אחוז הפקודות מתוך סך הפקודות שרצות	סוג הפקודה
35%	R-type
30%	Beq
20%	LW
15%	SW

נתונים זמני המחזור עבור שלוש הארכיטקטורות שנלמדו בקורס:

$T_{single\ cycle} = 6ns$

$T_{multicycle} = 4ns$

$T_{pipelined} = 4ns$

מעבד ה- Pipeline RISC-V בעל הנתונים הבאים:

- למעבד יחידות Forwarding **<u>מלא</u>** בין השלבים WB->Decode, Mem->Exe ו- WB->Exe, כנלמד בקורס.
- למעבד יחידת Hazard detection unit.
- החלטות על ביצוע הקפיצה מתקבלות בשלב ה- Exe.

נתון כי במחצית מפקודות ה-lw מתרחש Load hazard **וכי ב-50% מפקודות ה-beq לא מתבצעת קפיצה** (כלומר ב-50% כן מתבצעת קפיצה).

סמנו את התשובה הנכונה ביותר ביחס לזמן ריצת הקוד על גבי המעבדים השונים. בכל תשובה שם המעבד מייצג את זמן ריצת הקוד עליו.

הניחו שמספר הפקודות N גדול מאוד.

א. $pipelined < multi\ cycle < single\ cycle$

ב. $pipelined < single\ cycle < multi\ cycle$

ג. $pipelined < single\ cycle = multi\ cycle$

ד. $single\ cycle < multi\ cycle < pipelined$

ה. $pipelined = single\ cycle < multi\ cycle$

<u>פתרון</u>:

עבור מעבר ה-single cycle זמן ביצוע התוכנית הוא:

$$single\ cycle\ total\ run\ time = N \cdot 6ns$$

עבור מעבד ה-multicycle עלינו לעשות הפרדה לפי סוגי הפקודות:

$$multicycle\ total\ run\ time:$$

$$= N \cdot (0.35 \cdot 4 + 0.3 \cdot 3 + 0.2 \cdot 5 + 0.15 \cdot 4) \cdot 4ns = N \cdot 3.9 \cdot 4ns$$

עבור מעבד ה-pipeline נשים לב כי במחצית מפקודות ה-lw מתרחש hazard, ובכל פעם שזה קורה ירוץ מחזור ריק (bubble) על גבי המעבד. על פי הנתון קפיצות מותנות לא נלקחות רק במחצית מהפעמים ולכן יד תוספת של שני nops עבורן רק במחצית מהפעמים. מספר הפקודות שנריץ בסה"כ הוא:

$$N \cdot (1 + 0.1 + 0.15 \cdot 2) = N \cdot 1.4$$

בכדי לקבל את זמן הריצה עלינו להוסיף 4 מחזורי שעון לזמן זה (מילוי ה-pipe) ולהכפיל בזמן המחזור:

$$pipelined\ total\ run\ time = (N \cdot 1.4 + 4) \cdot 4ns \xrightarrow{4 \ll N} N \cdot 1.4 \cdot 4ns$$

כיוון ש-N זהה לכל סוגי המעבדים נוכל לצמצם אותו. לסיכום:

$single\ cycle = 6ns$

$multi\ cycle = 15.6ns$

$pipelined = 5.6ns$

$$pipelined < single\ cycle < multi\ cycle$$

תשובה ב'.

שאלה 13 (5 נקודות):

המהנדס ריק בחברת MortyCycle הוסיף פקודה חדשה למעבד RISC-V Multi Cycle:

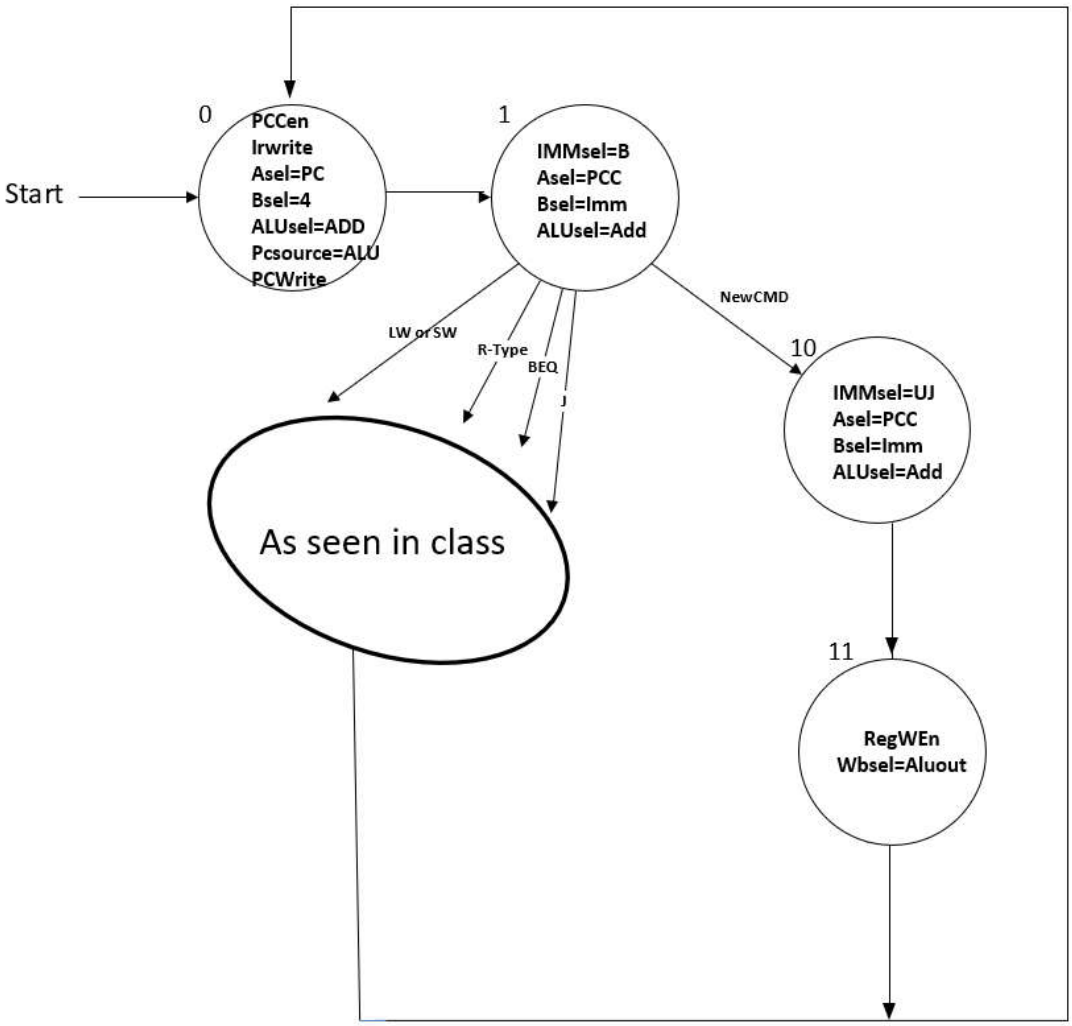
newCMD rd

הפקודה היא בפורמט UJ, ומקודדת כך:

0x000000	rd	opcode

על מנת לתמוך בפקודה, המהנדס שינה את מכונת המצבים של המעבד, והשאיר את ה- Datapath ללא שינוי.

מכונת המצבים המעודכנת הינה:



המהנדס מריץ את קטע הקוד הבא:

0x24: addi t8, x0,1

0x28: newCMD t8

המהנדס מעוניין גם לדעת האם ניתן לממש את הפקודה החדשה כפסאודו-פקודה של פקודות המכונה עליהן למד בקורס:
Jal, R-type, LW, SW, beq, addi

(תיאור הפקודות מופיע בדפי העזר)

סמנו את התשובה הנכונה:

א. הערך של t8 לאחר הרצת הקוד הינו 0x1.
ניתן לממש את הפקודה החדשה כפסאודו-פקודה שתתורגם לפקודת מכונה אחת.
ב. הערך של t8 לאחר הרצת הקוד הינו 0x24.
ניתן לממש את הפקודה החדשה כפסאודו-פקודה שתתורגם לפקודת מכונה אחת.
ג. הערך של t8 לאחר הרצת הינו 0x24.
ניתן לממש את הפקודה החדשה כפסאודו-פקודה שתתורגם לשתי פקודות מכונה, ולא ניתן לממש את הפקודה ע"י פחות פקודות מכונה.
ד. הערך של t8 לאחר הרצת הקוד יהיה 0x28.
ניתן לממש את הפקודה החדשה כפסאודו-פקודה שתתורגם לפקודת מכונה אחת.
ה. הערך של t8 לאחר הרצת הקוד יהיה 0x28.
ניתן לממש את הפקודה החדשה כפסאודו-פקודה שתתורגם לשתי פקודות מכונה, ולא ניתן לממש את הפקודה ע"י פחות פקודות מכונה.

פתרון:

תשובה ה'.

הפקודה החדשה מעדכנת את רגיסטר rd להיות כתובת הפקודה (כלומר Reg[rd]=PC).

שימו לב שניתן לממש את הפקודה החדשה כפסאודו פקודה של פקודות מכונה קיימות:

Jal rd next

next: addi rd, rd, -4

אי אפשר לממש את הפקודה ע״י פקודת מכונה אחת (כלומר רק ע"י JAL), משום שהפקודה newCMD צריכה לשמור ברגיסטר המתאים את ערך ה- PC של הפקודה הנוכחית. לכן, אם ננסה לעשות את זה בפקודה אחת אנחנו נתקע בלולאה אינסופית כי ה- PC לא מתקדם:

```
Jal rd, 0
```

פתרון אפשרי נוסף הינו שינוי ה- imm של ה- addi:

```
Jal rd next
next: addi rd, rd, 0
```

<u>שאלה 14 (5 נקודות):</u>

מהנדס בונה מערכת משדר מקלט מבוסס UART כנלמד בהרצאות. המשדר מבוסס על מעבד Pipeline-RISCV בעל הנתונים הבאים:

- למעבד יחידות Forwarding **<u>מלא</u>** בין השלבים WB->Decode, Mem->Exe ו- WB->Exe, כנלמד בקורס.
- למעבד יחידת Hazard Detection כנלמד בקורס.
- למעבד קיימת יחידת branch-prediction החוזה את יעד קפיצת פעולות branch כבר בשלב ה- fetch, **והחיזוי שלה תמיד נכון**.

נתון כי המשדר משדר את התוכן של כתובת הזיכרון $0x100$ כל מחזור שעון, **החל מהמחזור לאחר שהערך עודכן**.

בנוסף, נתון כי זמן מחזור השעון הינו $10ns$, וזמן T_{bit} הינו $100ns$.

נתון הקטע קוד הבא:

```
Main:   addi x5, x0, 0x100 // address of bit-to-send
        addi x6, x0, 0x300 // address of Array-to-send
        addi x4, x0, 1   // x4 holds 1 for stopbit
        sw x0, 0(x5) // Prepare start bit
        addi x28, x0, 8  // x28: number of data bits left
        // Insert nops here
Start:  lw x7, 0(x6) // Load new bit
        sw x7, 0(x5) // Prepare new bit
        // Insert nops here
        addi x6, x6, 4 // Advance to next bit
        addi x28, x28, -1
        bne x28, x0, start
        // Insert nops here
        sw x4, 0(x5) // Prepare stop bit
        10 X nop
        // rest of code
```

בתוכנית ישנם סימונים שבהם יש מקום להוספת *nop* כדי לעמוד בתנאים (אין חובה להוסיף *nop* בכל מקום).

רמז – מטרת הכנסת ה- *nop* היא יצירת T_{bit} באורך תקין.

כמה *nop* יש להוסיף (לא צריך להתחשב בתשובות ב- 10 ה- *nop* שמופיעים בשורה 15)?

א. 9

ב. 10

ג. 11

ד. 12

ה. 13

פתרון:

תשובה ד'.

נשים לב כי יש *load-hazard* בין פקודה מספר 7 ובין פקודה מספר שמונה, לכן יש שמה *bubble* יחיד. בנוסף, נתונה יחידת חיזוי מושלמת, ולכן אין *flush*. לכן, נצטרך להוסיף *nop* כמפורט:

```
1   addi x5, x0, 0x100
2   addi x6, x0, 0x300
3   addi x7, x0, 0x8
4   sw x7, 0(x5)
5   addi x28, x0, 8
6   // 6 nops
7   start: lw x7, 0(x6)
8   sw x7, 0(x5)
9   // 4 nops
10  addi x6, x6, 4
11  addi x28, x28, -1
12  bne x28, x0, start
13  // 2 nops
14  sw x0, 0(x5)
15  10 x nops
16  // rest of code
```

סה"כ 12 *nops*.

<u>הסבר מפורט</u>: לפי הנתון, הערך מתחיל שידור במחזור לאחר שהערך מעודכן בזכרון. אם פקודה מעדכנת את הערך בזכרון, אזי הוא מעודכן בשלב ה-*MEM*, ולכן שלב ה-*WB* הינו המחזור הראשון.

1. הפקודה הראשונה שמשנה את הערך הינו פקודה 4. הפקודה הבאה הינה פקודה 8 – נוודא שיש 10 מחזורים בין שלב ה-*MEM* של פקודה 4 ושלב ה-*MEM* של פקודה 8 (כולל פקודה 8

עצמה). ישנם 3 פקודות (פקודות 5 7 ו-8), ויש *load hazard* אחד, ולכן נצטרך להוסיף 6 פקודות *nop* בין לבין במקום הנתון.

2. הפקודה הבאה שמשנה הינה פקודה 8, ולאחריה פקודה 8 הבאה בלולאה. ביניהם יש 5 פקודות (10 11 12 7 8), ו-*load hazard* אחד, ולכן נצטרך 4 *nop*. אין *flush* כי נתון מנגנון חיזוי מושלם.
3. הפקודה הבאה שמשנה הינה פקודה 8 ולאחריה 14 (כאשר כבר לא בלולאה). ביניהם 4 פקודות (10 11 12 14), 4 *nop* בשורה 9, ולכן נצטרך להוסיף עוד 2 *nop*.

<u>החל מהעמוד הבא מתחיל החלק של השאלות פתוחות</u>

<u>(שאלות 15 – 17)</u>

<u>שאלה 15 (10 נקודות):</u>

בשאלה הזאת נתון קטע הקוד הבא הכתוב ב assembly עם הכתובת של כל פקודה:

```
Address
0x1000   function:  li t0, 0
0x1004              addi t1, a0, 0
0x1008   loop:      bge t0, a1, end
0x100C              mul a0, a0, t1
0x1010              addi t0, t0, 1
0x1014              jal x0, loop
0x1018   end        jr ra
```

בנוסף נתון כי מאתחלים את הרגיסטרים באופן הבא:

$$pc = 1000$$

$$a0 = 2$$

$$a1 = 5$$

$$t0 = 0$$

$$t1 = 0$$

$$ra = address\ of\ the\ end\ of\ the\ code$$

א. כמה פעמים הקוד מבצע את הפקודה בכתובת 1010 ?

ב. כעת נתון a0=x, a1=y בתחילת הריצה.
מה הערך של הרגיסטר $a0$ בסוף ריצת הקוד כפונקציה של x ו- y?

ג. מבצעים את השינוי הבא בקוד: הפקודה בכתובת 0x1014 מוחלפת בפקודה j loop.

איך השינוי הזה ישפיע על הקוד (התחשבו רק בקוד הנתון) ? נמקו.

<u>פתרון</u>:

התוכנית הנתונה היא פונקציה רקורסיבית המחשבת את הערך של $a_0^{a_1+1}$

מכיוון שהערך ההתחלתי של רגיסטר a_1 הוא 5, הקוד יבצע 5 פעמים את הלולאה ולכן בפרט יבצע 5 פעמים את הפעולה בכתובת 1010.

בסוף ריצת התוכנית, הערך של a_0 הוא $a_0 = x^{y+1}$

בסעיף 3 השינוי שאנחנו מבצעים לא משפיע על הקוד הנתון מכיוון שפעולת jal שומרת את ערך הכתובת שממנה קפצנו על מנת שנוכל לחזור אליה. כאן אין צורך בכתובת הזאת ולכן השינוי לא משפיע על הקוד.

<u>שאלה 16 (2 נקודות לכל סעיף):</u>

נתונה מערכת עקיבה (FSM) <u>מסוג מילי</u> בעלת N מצבים (N לא ידוע) עם כניסה יחידה X, ומוצא יחיד Z.

דני ויוסי מצאו סדרות הפרדה שונות המתחילות משני מצבים כלשהם A ו- B של המערכת, עבורן המערכת מוציאה פלט שונה בסוף הסדרה. פלט מכונת המצבים במהלך הסדרות לא ידוע.

הסדרה שדני מצא באורך 12 קלטים והסדרה שיוסי מצא באורך 8 קלטים.

לגבי כל אחד מהמשפטים בטבלה להלן, סמנו X רק בעמודה המתאימה.
העמודות מסמלות אם המשפט נכון, לא בהכרח נכון (כלומר, ניתן למצוא דוגמה בה מתקיים המשפט ודוגמה בה הוא לא מתקיים) או בהכרח לא נכון.

		נכון	לא בהכרח נכון	בהכרח לא נכון
1.	ייתכן ש- N=2			
2.	קיימת סדרת הפרדה נוספת, באורך 8 סיביות או קצרה יותר, השונה מסדרתו של יוסי			
3.	ייתכן ש-8 הכניסות הראשונות של הסדרה של דני זהות לכניסות של סדרתו של יוסי			
4.	ניתן למצוא סדרות הפרדה בכל כפולה של 8 כניסות (סדרה באורך 16, 24,...)			
5.	אם N=8, קיימת סדרת הפרדה קצרה יותר מזו של יוסי			

פתרון:

1 – התקבלו התשובות "נכון" או "לא בהכרח נכון". המכונה יכולה להיות בעלת 2 מצבים בלבד. למשל אם יש שני מצבים, מצב A שמוציא 0 עבור קלט 0 ועובר לעצמו, ועבור קלט 1 עובר למצב B ומוציא 1, ומצב נוסף B שעובר לעצמו ומוציא 0 לכל קלט, אז כל סדרה של אפסים שבסופה יש '1' תוכל להיות סדרת הפרדה כנתון בשאלה.

2 – לא בהכרח נכון. יכול להיות שהסדרה של יוסי היא סדרת ההפרדה הקצרה ביותר.

3 – התקבלו התשובות "נכון" או "לא בהכרח נכון". אנחנו לא יודעים מה פלט המכונה במהלך סדרת ההפרדה.

4 – לא בהכרח נכון. יכול להיות שמכונת המצבים נכנסת למצב שעובר רק לעצמו לאחר 13 מצבים.

5 – בהכרח נכון. לפי המשפט הנלמד בהרצאה: אם Si ו- Sj שני מצבים בני הפרדה במכונה M בעלת N מצבים, אז קיימת סדרת-הפרדה באורך של N-1 לכל היותר.

<u>שאלה 17 (10 נקודות):</u>

במעבד Multicycle RISCV הוחלט לטפל בשני סוגי חריגות בלבד: חלוקה באפס ופקודה לא חוקית, עם הקידוד הבא:

סוג החריגה	קידוד
פקודה לא חוקית	1
חלוקה באפס	2

במקרה של חלוקה באפס צריך לכתוב לרגיסטר המכנה את הערך 1.
במקרה של פקודה לא חוקית צריך לדלג על הפקודה הלא חוקית ולהמשיך לפקודה הבאה.

הניחו שכל הרגיסטרים מסוג s_i, t_i ו- a_i, מאותחלים ל- 0 ושהגישה לרגיסטרים SCAUSE ו- SEPC זהה לגישה לרגיסטר מה- RegFile.
התוכנית רצה החל מ- main.

השלימו את הקוד בעמוד הבא כך שירוץ באופן תקין ויתמוך בחריגות הנתונות בשאלה.

```
0x10000000 main:      addi t0, x0, 2
0x10000004             sub t2, t1, t1
0x10000008            unknown instruction
0x1000000C            div t3, t0, t2

0x1C090000 interrupt handler:   addi sp, sp, _______
0x1C090004                      sw s0, 0(sp)
0x1C090008                      sw _____, 4(sp)
0x1C09000C                      addi s0, x0, 1
0x1C090010                      addi s1, x0, _____
0x1C090014                      beq SCAUSE, s0, _______
0x1C090018                      beq SCAUSE, ____ , label2


0x1C09001C done:                lw s0, 0(sp)
0x1C090020                      lw_______, 4(sp)
0x1C090024                      addi sp, sp ,8
0x1C090028                      jr _______
0x1C091000 label1:              addi _______, _______, 4
0x1C091004                      j done
0x1C091008 label2:              addi t2, x0, _____
0x1C09100C                      j done
0x1C091008 label:
```

פתרון:

```
0x10000000 main:        addi t0, x0, 2
0x10000004               sub t2, t1, t1
0x10000008              unknown instruction
0x1000000C              div t3, t0, t2

0x1C090000 interrupt handler:   addi sp, sp, -8
0x1C090004                      sw s0, 0(sp)
0x1C090008                      sw s1, 4(sp)
0x1C09000C                      addi s0, x0, 1
0x1C090010                      addi s1, x0, 2
0x1C090014                      beq SCAUSE, s0, label1
0x1C090018                      beq SCAUSE, s1, label2

0x1C09001C done:                lw s0, 0(sp)
0x1C090020                      lw s1, 4(sp)
0x1C090024                      addi sp, sp ,8
0x1C090028                      jr SEPC
0x1C091000 label1:              addi SEPC, SEPC, 4
0x1C091004                      j done
0x1C091008 label2:              addi t2, x0, 1
0x1C09100C                      j done
```