מערכות ספרתיות ומבנה המחשב (044252)

סמסטר חורף תשע"ט

בחינה סופית – מועד ב

4 במרץ 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.

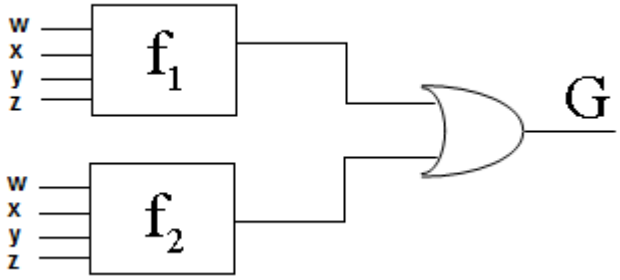
הנחיות והוראות:

- הבחינה כתובה על גבי 20 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 – הבהוב סטטי (5 נקודות)

נתון המעגל הבא



נתונות הפונקציות

$$f_1(w,x,y,z) = (w+z)\cdot(w'+x'+z)\cdot(x'+y'+z)$$

$$f_2(w,x,y,z) = w'xy + wy + wx' + xyz$$

הפונקציה f_1 ממומשת כמכפלת סכומים, בעזרת 3 שערי OR חלקם עם 3 כניסות, שער AND אחד בעל 3 כניסות ושערי NOT.
הפונקציה f_2 ממומשת כסכום מכפלות, בעזרת 4 שערי AND חלקם עם 3 כניסות, שער OR אחד בעל 4 כניסות ושערי NOT.

<u>הבהוב סטטי מסוג LL</u> קורה כאשר משנים כניסה אחת, היציאה אמורה להיות סטטית ב-0 אבל היא משתנה ל-1 באופן רגעי.
<u>הבהוב סטטי מסוג HH</u> קורה כאשר משנים כניסה אחת, היציאה אמורה להיות סטטית ב-1 אבל היא משתנה ל-0 באופן רגעי.

מבין הטענות הבאות שמתייחסות לנקודה G, בחרו את הטענה הנכונה החזקה ביותר

א- **<u>ייתכן</u>** הבהוב סטטי מסוג LL אך **<u>לא ייתכן</u>** הבהוב סטטי מסוג HH
ב- **<u>לא ייתכן</u>** הבהוב סטטי מסוג LL אך **<u>ייתכן</u>** הבהוב סטטי מסוג HH
ג- **<u>ייתכן</u>** הבהוב סטטי מסוג LL **<u>וייתכן</u>** הבהוב סטטי מסוג HH
ד- **<u>לא ייתכן</u>** הבהוב סטטי במעגל הנתון מכל סוג
ה- מנתוני השאלה, לא ניתן לדעת אם ייתכן הבהוב סטטי

פתרון: תשובה ג'

$\frac{wx}{yz}$	00	01	11	10
00	0	0	0	
01				
11				
10	0	0	0	

OR

$\frac{wx}{yz}$	00	01	11	10
00				1
01				1
11		1	1	1
10		1	1	1

ו-

עבור השינוי $wxyz: 0100 \rightarrow 1100$ ייתכן הבהוב סטטי מ-0 ל-0 בפונקציה f_1, הערך של הפונקציה f_2 עבור הצירוף זה הוא 0 ולכן זה יכול לגרום להבהוב מסוג LL בנקודה G.
עבור השינוי $wxyz: 0110 \rightarrow 1110$ ייתכן הבהוב סטטי מ-1 ל-1 בפונקציה f_2, הערך של פונקציה f_1 עבור הצירוף זה הוא 0 ולכן זה יכול לגרום להבהוב מסוג HH בנקודה G.

שאלה 2 – SC-RISCV (5 נקודות)

לאחר ייצור מעבד *Single Cycle RISCV*, התומך בפקודות *beq, bne* ו*ble* התגלתה תקלה ברכיב ה- *Branch Comparator*.
עובד חרוץ גילה שכאשר מחליפים את הכניסות לרכיב, כלומר מכניסים לכניסה *A* של ה- *Branch comparator* את רגיסטר *rs2* ולכניסה *B* את רגיסטר *rs1*, הרכיב עובד בצורה תקינה בכל הפקודות הנתמכות מסוג *branch*.
שימו לב - פקודת *ble* הינה פקודת *branch lower equal*.
איזה מהשינויים הבאים יפתור בוודאות את הבעיה עבור כל פקודות ה*branch* הנתמכות, בהנחה שלא מחליפים את הכניסות של ה- *Branch Comparator* כפי שהציע העובד?
נדרשת התשובה החזקה ביותר מבין הנכונות (לדוגמה, ה' גוברת על ג' אם שתיהן נכונות).

א- הפיכת המוצא *BrEq* בלבד.
ב- הפיכת סיבית הבקרה של ה- *PCselector* בלבד.
ג- הפיכת סיבית המוצא *BrEq* וסיבית המוצא *BrLt*.
ד- הפיכת המוצא *BrLt* בלבד.
ה- גם אופציה ג' וגם אופציה ד' יפתרו את התקלה.

פתרון: תשובה ד'
מכיוון שכל התקלה היא היפוך הכניסות, המוצא שקובע האם הכניסות שוות עדיין יעבוד בצורה תקינה ואין צורך להפוך אותו.

שאלה 3 - MC-RISCV (5 נקודות)

למעבד *Multi Cycle RISCV* כפי שנלמד בכיתה, רוצים לכתוב פקודה חדשה, *add3ri*, שתעשה שימוש ב- 3 רגיסטרים ו- *immediate* בגודל 12 סיביות. הפקודה תבצע חיבור של 2 רגיסטרים + *immediate* ותשמור את התוצאה ברגיסטר שלישי. הפקודה נראית כך:

add3ri rd, rs1, rs2, imm : R[rd] = R[rs1] + R[rs2] + imm

מכיוון שיש רק *32* סיביות לייצוג הפקודה, לא היה מקום לציון מספרי שלושת הרגיסטרים ולכן הוחלט שבכל שימוש בפקודה יתקיים- *rd = rs1*. כלומר שהרגיסטר *rs1* שמכיל מידע הוא גם הרגיסטר שרושמים אליו את התוצאה. הדוגמא הקודמת תיראה כך:
add3ri rs1, rs2, imm : R[rs1] = R[rs1] + R[rs2] + imm

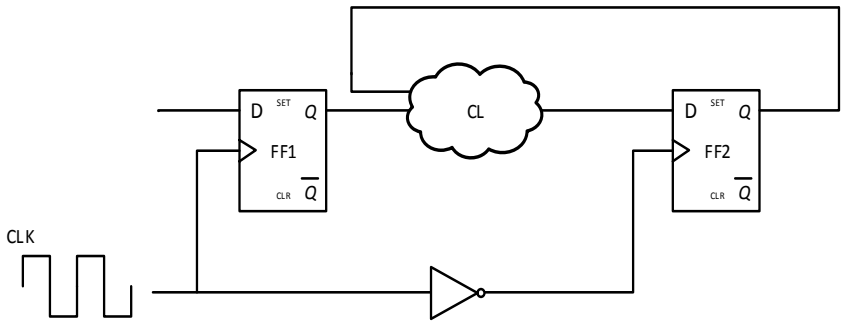
האם ניתן לממש פקודה זו במעבד תוך המשך תמיכה בשאר הפקודות?

א- לא. מכיוון שלא ניתן לחבר 3 גורמים בבת אחת בעזרת *ALU* בודד, לא ניתן לממש את הפקודה במעבד הנתון.
ב- כן. ניתן ע"י הוספת בוררים ואותות בקרה ושינוי מכונת המצבים בלבד.
ג- כן. אין צורך בהוספת בוררים ואותות בקרה אך יהיה צורך בהגדלת אחד הבוררים הקיימים ובהוספת מצבים למכונת המצבים.
ד- כן, אך נהיה חייבים לבצע שינוי ברכיב ה-*RegFile*.
ה- לא. אין שינויים ב-*datapath* או בבקר שיאפשרו תמיכה בפקודה, אך ניתן לממש אם גודל ה-*immediate* בפקודה יהיה מוגבל להיות עד 7 ביטים.

פתרון: תשובה ב'
יש כמה דרכים לפתור את הבעיה. נציג אחת מהן (הפשוטה מביניהן):
כדי לוודא שבשלב ה*DEC*, הכניסות *AddrA* ו- *AddrB* מקבלות את *rs1* ו-*rs2* בהתאמה ושקיים גם *immediate*, נבחר בפורמט של פקודת ה- *SW* – כלומר *S-format*.
במצב זה, כדי להעביר את מה שרשום ב*rs1* גם לכניסת ה*AddrD* (שאליה בדרך כלל מקושר ה*rd*) נהיה חייבים להשתמש בבורר שיבחר ב*rs1* עבור הפקודה הזו. בשאר הפקודות הבורר יבחר ב*rd* כרגיל. הבורר יכנס לפעולה בשלב ה*WB*. הפקודה דומה לפקודת *addi* אך צריך לבצע את שלב ה*EXE* פעמיים. בפעם הראשונה נחבר את- *rs1* עם ה*imm* ונשמור ב-*ALUout*. בפעם השנייה, נחבר את ה-*ALUout* עם *rs2* (ששמור ברגיסטר *B*).
לשם כך, עלינו להגדיל את הבורר *Asel* כדי שיכיל גם אופציה של *ALUout*. בשלב האחרון נשמור את הערך של *ALUout* בתוך ה*RegFile* כפי שתואר קודם. סה"כ צריך 5 שלבים לפקודה. כמובן שצריך לשנות את מכונת המצבים בהתאם.
בעקבות ההסבר הנ"ל, תשובה א' נפסלת.
כדי לממש את הפקודה כפי שנדרש, צריך שהערך *rs1* בפקודה יגיע גם לכניסה *AddrA* וגם לכניסה של *AddrD* ב- *RegFile*. אנו זקוקים לבורר לפחות על אחת מהכניסה הללו. לכן תשובה ג' נפסלת. אין שינוי במכונת המצבים שמאפשר להעביר ערך אחר ל*AddrD*.
תשובה ד' פשוט אינה נכונה כי אין בעיה לכתוב לרגיסטר (בשלב ה*WB*) שגם קוראים ממנו (בשלב ה*DEC*).
מכיוון שאנו בפועל משתמשים בכתיבת הפקודה ב2 רגיסטרים ו-*immediate*, כמו פקודת -*S type* רגילה, גם גודל ה *immediate* יכול להיות כמו בפקודת *S-type* והוא 12 ביטים. לכן תשובה ה' אינה נכונה.

שאלה 4 – זמני השהיה (5 נקודות)

נתון המעגל הבא



נתון כי הדלגלים דוגמים בעליית שעון. בנוסף, נתון כי כניסת המערכת מתוזמנת כך שהיא עומדת בדרישות ה- *hold* וה- *setup* של הדלגלג *FF1*, וכן כי **השעון סימטרי**.

להלן זמני ההשהיה:

$t_{pd}(CL) = 5\ ns$	$t_{cd}(CL) = 2\ ns$
$t_{pd}(not) = 2\ ns$	$t_{cd}(not) = 1\ ns$
$t_{pCQ}(FF) = 7\ ns$	$t_{cd}(FF) = 2\ ns$
$t_{hold}(FF) = 5\ ns$	$t_{setup}(FF) = 3\ ns$

מבין האפשרויות הבאות, מהו אורך מחזור השעון המינימלי שיאפשר למעגל לפעול בצורה תקינה?

א- $28\ ns$

ב- $29\ ns$

ג- $31\ ns$

ד- $33\ ns$

ה- המעגל לא יעבוד בצורה תקינה עם כל זמן מחזור שנבחר.

פתרון: תשובה ה'
במסלול מדלגלג 2 לעצמו תנאי *HOLD* לא מתקיים ולכן המעגל לעולם לא יעבוד בצורה תקינה עם כל זמן המחזור שנבחר

שאלה 5 – צמצום מכונת מצבים (5 נקודות)

בתהליך הצמצום של טבלת המצבים של מערכת עקיבה סינכרונית שיש לה כניסה אחת ויציאה אחת, התקבלה התמונה הבאה בתום השלב **השלישי**:

$$P_3 = (A)(BC)(D)(EF)$$

מבין התשובות הבאות, בחרו את התשובה הנכונה החזקה ביותר:

א- אם המערכת היא מסוג **Mealy** אז ב-P_1 יש **בדיוק 2** מחלקות שקילות

ב- אם המערכת היא מסוג **Mealy** וב-P_1 יש **יותר מ- 2** מחלקות שקילות אזי במכונה המצומצמת יש **לכל היותר** 4 מצבים

ג- אם המערכת היא מסוג **Moore** אז ב-P_1 יש **בדיוק 2** מחלקות שקילות

ד- תשובות א' ו- ג' נכונות

ה- תשובות ב' ו- ג' נכונות

פתרון:
תשובה נכונה ה'
תשובה ב' נכונה (4 מצבים בדיוק):

$$P_0 = (ABCDEF)$$
$$P_1 = (A)(BC)(DEF)$$
$$P_2 = (A)(BC)(D)(EF)$$
$$P_3 = P_2$$

ב-P_1 יש 3 מחלקות לכל הפחות, אם נבחר לחלק אותה ל-4 אז $P_3 = P_2 = P_1$ ועדיין נקבל 4 מצבים. (אפשר לראות שעבור כל חלוקה אחרת עדיין נקבל ש- $P_3 = P_2$).
תשובה א' לא נכונה, דוגמה נגדית - התשובה לעיל.
תשובה ג' נכונה, בכל שלב K כך ש- $P_{K-1} \neq P_K$ יש לפחות $K + 1$ מחלקות.
(*) מכיוון שהמערכת היא מסוג *Moore* עם יציאה אחת, אז אנחנו יודעים שב- P_1 יכולים להיות לכל היותר 2 מחלקות שקילות.

$$|P_3| = 4 \geq |P_2| \geq |P_1| > |P_0| = 1 \quad \rightarrow \quad |P_1| > |P_0| \quad \rightarrow \quad P_1 \neq P_0$$

מצד אחד $|P_1| > 1$ ומצד שני לפי (*) $|P_1| \leq 2$ ולכן $P_1 = 2$

שאלה 6 – Verilog (5 נקודות)

נתון כי הסיגנל a הכיל את הערך 0x1234 לפני עליית השעון. עבור אילו מקטעי הקוד הבאים יכיל הסיגנל a בהכרח את הערך 0x3412 לאחר עליית השעון? בחרו את התשובה הנכונה:

א-

```
always @(posedge clk) begin
   a2 <= a;
   a[7:0] <= a[15:8];
   a[15:8] <= a2[7:0];
end
```

ב-

```
always @(posedge clk) begin
      a[7:0] <= a[15:8];
      a[15:8] <= a[7:0];
End
```

ג-

```
always @(posedge clk) begin
   a2 = a;
   a[7:0] = a[15:8];
   a[15:8] = a2[7:0];
end
```

ד- תשובות א', ב' נכונות
ה- תשובות ב', ג' נכונות

פתרון: תשובה ה'
נשים לב שבתשובה א' ההשמה מסוג non-blocking, לכן צד ימין של כל המשוואות יחושב שורה אחרי שורה, אך רק בסוף ה-process ההשמות יתבצעו (במקביל). בחישוב הביטוי של ההשמה השלישית לא ניתן לדעת מה יש ב-a2 (לא נתון) ולכן התשובה אינה נכונה.
בתשובה ב' ההשמה גם היא מסוג non-blocking, אך כאן משתמשים בערך של a ישירות בכל ההשמות. מכיוון שהביטויים מחושבים לפני שההשמות מתבצעות, הביטים התחתונים יתחלפו עם העליונים ולהפך ללא דריסת המידע, כדרוש. לכן תשובה ב' נכונה.
בתשובה ג' ההשמה היא blocking. במקרה זה, עוברים לשורה הבאה רק לאחר שההשמה התבצעה. לאחר ביצוע השורה הראשונה a2 מקבל את הערך 0x1234. לאחר מכן, הביטים התחתונים של a נדרסים ע"י הביטים העליונים. לאחר ביצוע ההשמה הזו הערך של a יהיה 0x1212. לבסוף, הביטים העליונים של a נדרסים ע"י הביטים התחתונים המקוריים, שאוחסנו ב-a2 (שכבר מכיל 0x1234 בזכות סוג ההשמה). סה"כ a יקבל את הערך 0x3412 וגם תשובה זו נכונה.

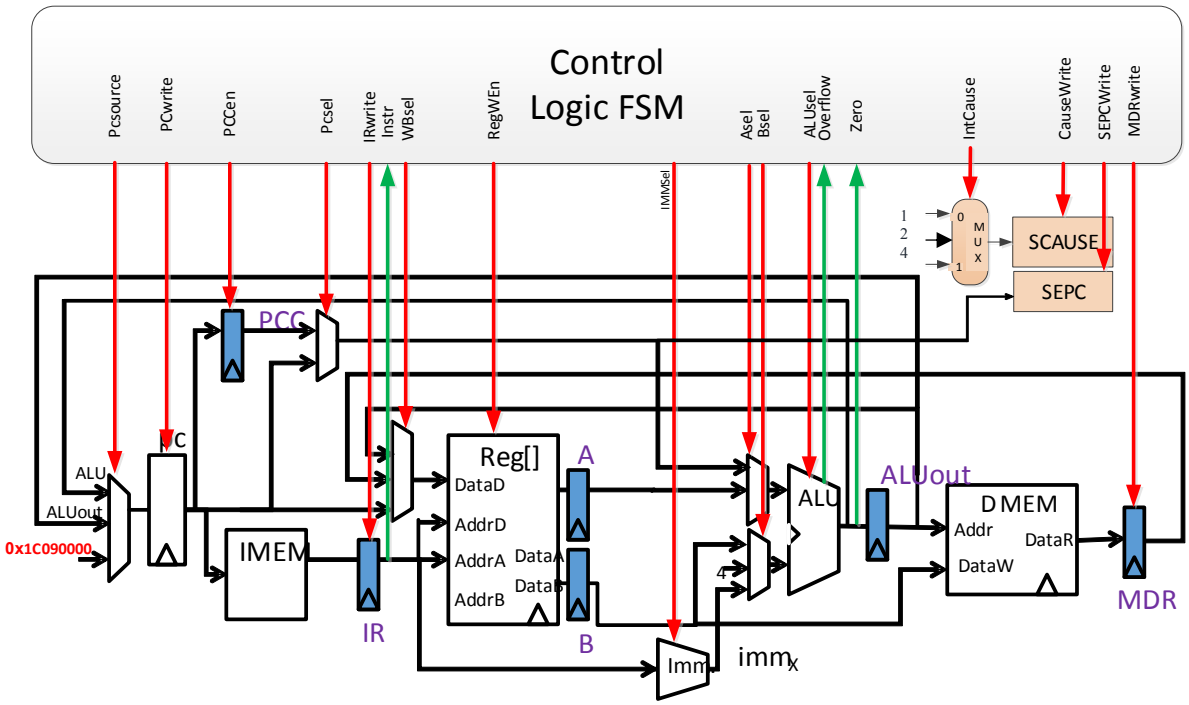
שאלות 7-8 – פסיקות

במעבד Multi Cycle RISCV הוחלט לטפל ב-3 סוגים של חריגות בלבד, לפי שיטת "קוד הגורם לחריגה": גלישה, חילוק באפס ופקודה לא חוקית.

שיטת קידוד קוד החריגות שמשתמשים בה היא "חזקות של 2", באופן הבא:

סוג החריגה	קידוד
גלישה	1
חילוק באפס	2
פקודה לא חוקית	4

להלן המימוש של Multi-Cycle RISCV עם תמיכה בפסיקות



תזכורת הערך הנטען לרגיסטר SEPC לפני הקריאה לפונקציית הטיפול בחריגה הוא PCC

הניחו **שכל הרגיסטרים מאותחלים ל-0** ושהגישה לרגיסטרים SCAUSE ו- SEPC היא גישה לרגיסטר רגיל. התוכנית רצה החל מהפונקציה main:

```
0x10000000      main:       addi t0, x0, 8
0x10000004                  addi s1, x0, 1
0x10000008                  div t2, t0, t3
0x1000000C                  add t2, t2, t2
0x10000010      exit:

0x1C090000 interrupt handler: addi sp, sp, -4
0x1C090004                  sw s0, 0(sp)
0x1C090008                  addi s0, x0, 1
0x1C09000C                  addi s1, x0, 2
0x1C090010                  beq SCAUSE, s0, label1
0x1C090014                  beq SCAUSE, s1, label2
0x1C090018                  beq SCAUSE, s1, label3
0x1C09001C      done:       lw s0, 0(sp)
0x1C090020                  addi sp, sp ,4
0x1C090024                  jr SEPC

0x1C091000      label1:     addi t3, t3, 1
0x1C091004                  j ret
0x1C091008      label2:     add t3, t3, s1
0x1C09100C                  j ret
0x1C091010      label3:     add t3, t3, s0
0x1C091014      ret:        j done
```

שאלות 7 (5 נקודות)

מהו הערך שיתקבל ברגיסטר **s1** **אחרי** ביצוע הפקודה div בכתובת 0x10000008 ?

א- 0
ב- 1
ג- 2
ד- 4
ה- לעולם לא תתבצע הפקודה בכתובת 0x10000008

פתרון: תשובה נכונה ג'
שימו לב שפונקציית הטיפול בחריגה משנה את הערך של הרגיסטר s1 מבלי לגבות אותו ולכן הערך שלו ישתנה ל- 2

שאלה 8 (5 נקודות)

מהו הערך שיתקבל ברגיסטר **t2** **אחרי** ביצוע הפקודה add בכתובת 0x1000000C?

א- 8
ב- 7
ג- 6
ד- 5
ה- לעולם לא תתבצע הפקודה בכתובת `0x1000000C`

פתרון תשובה א'

פקודת החילוק תגרום לחריגה מסוג "חילוק באפס".
SCAUSE=2, SEPC= 0x10000008
קופצים לכתובת הקבועה 0x1C090000 לטיפול בפסיקות, מגבים את הרגיסטרים s0 אך לא את s1 למרות שהולכים לשנות אותו. קופצים ל-label2. הערך של t3 הוא 0, משנים אותו ל-2 וחוזרים לפונקציית הטיפול בחריגות שמשחררת את המחסנית ומחזירה אותנו לבצע **שוב את הפקודה שגרמה לחריגה** כאשר t3=2 במקום 0, ולכן הפעם במהלך ביצוע הפקודה לא תתרחש חריגה ואכן נצליח לסיים את התוכנית, לאחר ביצוע פעולת החילוק:

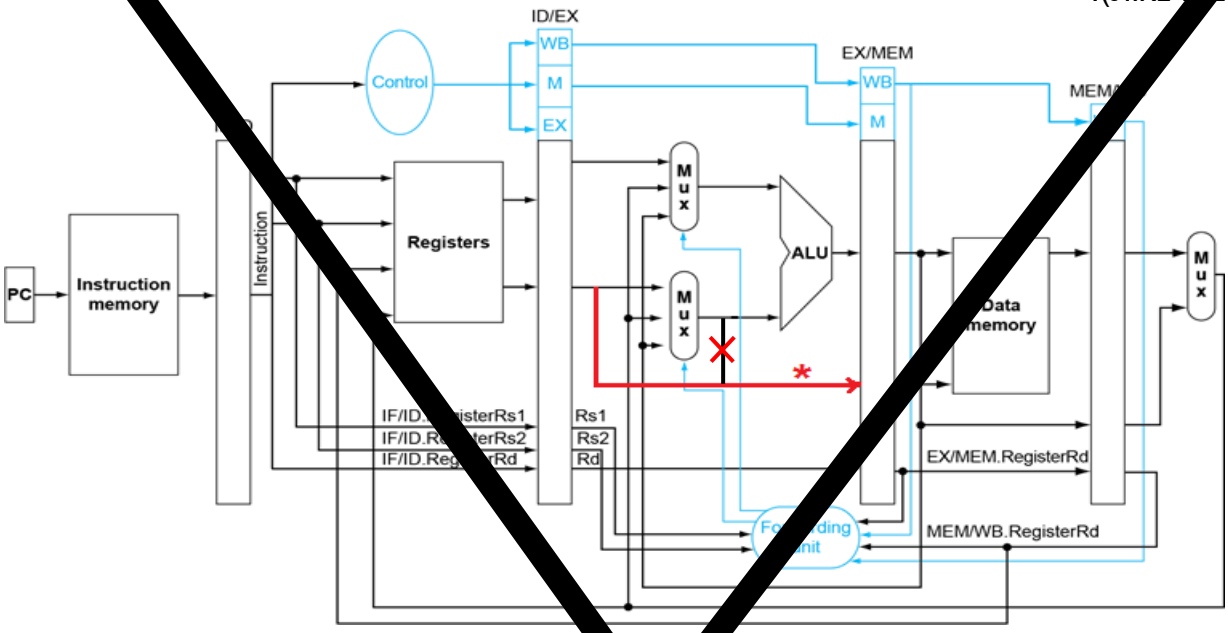
$$t2 = \frac{t0}{t3} = \frac{8}{2} = 4$$

ואחרי פעולת החיבור: $t2 = t2 + t2 = 4 + 4 = 8$

השאלה אינה תקינה. גרסה תקינה ומורחבת של השאלה מופיעה בעמודים הבאים (לקוח ממצגות הזום שבמודל)

שאלה 9 – Pipelined RISCV (5 נקודות)

שאלה זו מתייחסת ל- *Pipelined RISCV* הבא (המכיל חלקים רלבנטיים מזה שנלמד בהרצאות):



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*
- המעבד מניח תמיד שפקודות *branch* לא נלקחות, ומכיל מנגנון *Flushing* במקרה שכן. ההחלטה הסופית על קפיצות מתבצעת בתוך שלב ה-*EXE*.

הערה: שימו לב לחוט המסומן ב-*X*. במימוש המקורי של ה- *Pipeline RISCV*. במעבד כפי שנלמד, החוט הזה היה מחובר למוצא של ה- *MUX* (מסומן ב-*X*). במימוש החדש ניתקנו אותו מהמוצא של ה-*MUX* וחיברנו אותו למקום אחר כמו שמתואר בציור.

נתון קטע הקוד הבא:

```
lw t1, 0(s1)
sw t1, 0(s0)
```

מה המספר המינימלי של פקודות *NOP* שיש להוסיף לקוד לעיל כדי שירוץ כשורה?

א- 0
ב- 1
ג- 2
ד- 3
ה- 4

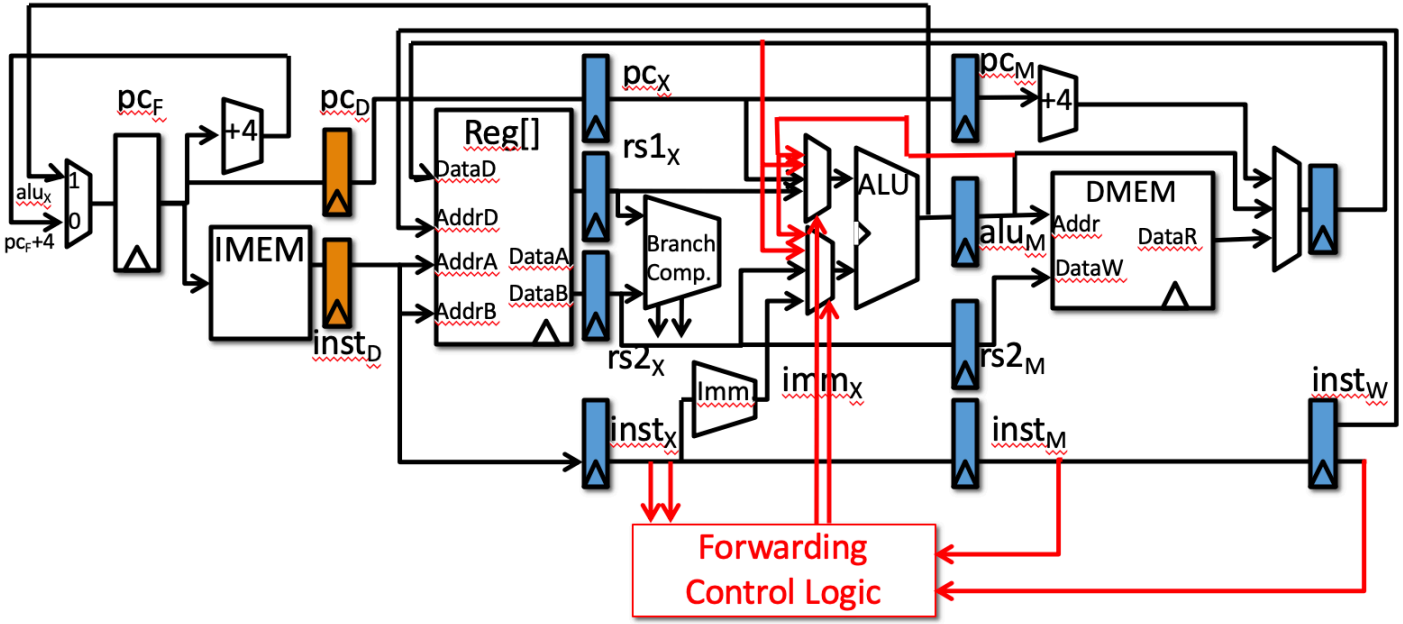
פתרון: תשובה ד'

צריך להוסיף 3 פקודות *NOP*: *NOP* ראשון מכיוון שאין יחידת *Hazard Detection* ועוד שני *NOP*-ים מכיוון שלא ניתן לעשות *forwarding* לערך שנקרא מהזיכרון בפקודת ה-*lw* לרגיסטר אותו רוצים לכתוב בפקודת *sw*. ולכן צריך לחכות עד שפקודת ה-*lw* תסיים את שלב ה-*WB* - פקודת *sw* בינתיים תחכה בשלב ה- *decode*. שימו לב שמספר ה-*NOP*-ים לא היה משתנה גם אם לא היינו משנים את החיבור של החוטים בשלב ה-*EXE*.

שאלה 3א – חורף תשע"ט מועד ב' (גרסה מתוקנת ומורחבת)

נתון מעבד Pipelined RISC-V כפי שנלמד בתרגול

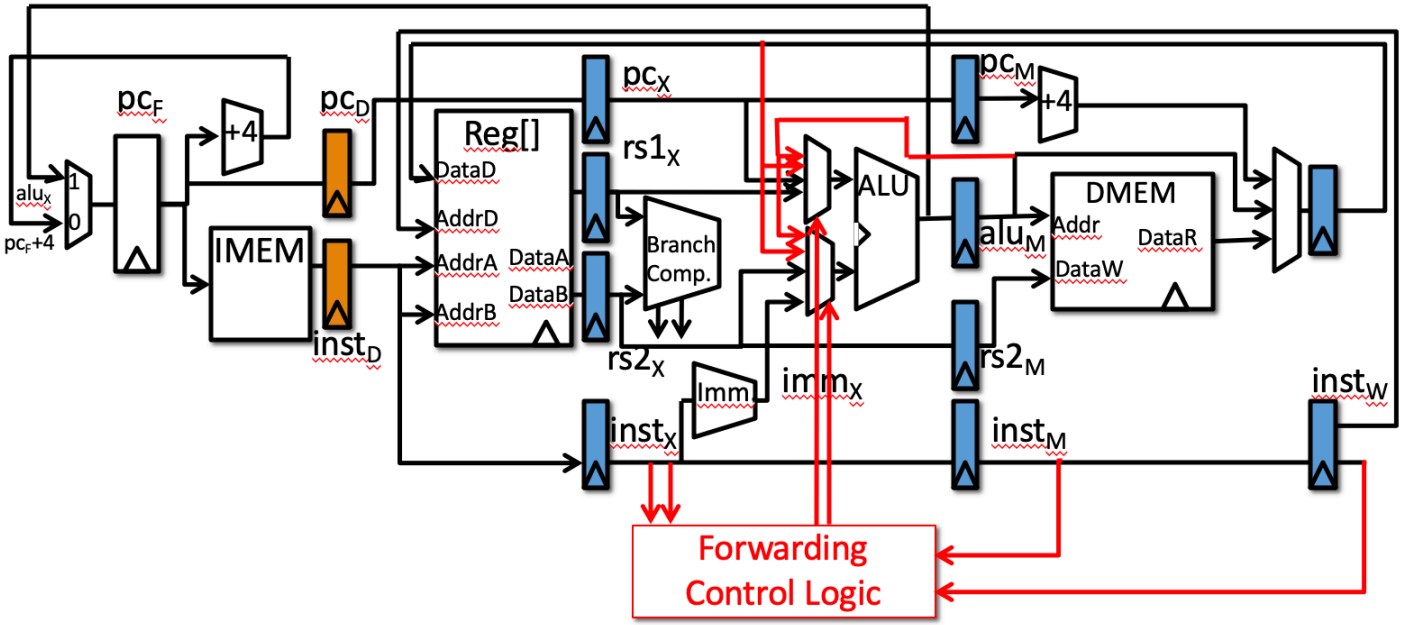
- **קיים** *Forwarding* מלא, אך **לא קיים** *Forwarding* בתוך ה- *Register File* (כלומר – לא קיים *Forwarding* בין שלב ה-*WB* לשלב ה-*Decode*).
- **לא קיימת** יחידת *Hazard Detection*
- המעבד מניח תמיד שפקודות *branch* לא נלקחות, ומכיל מנגנון *Flushing* במקרה שכן. ההחלטה הסופית על קפיצות מתבצעת בתום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



סעיף חימום (לא הופיע במבחן) – בהינתן הקוד להלן, מה המספר המינימלי של פקודות NOP שנצטרך להוסיף בין הפקודות כדי שהקוד ירוץ כשורה?

LW t1, 0(s1)
SW t2, 0(t1)

א- 0
ב- 1
ג- 2
ד- 3
ה- 4

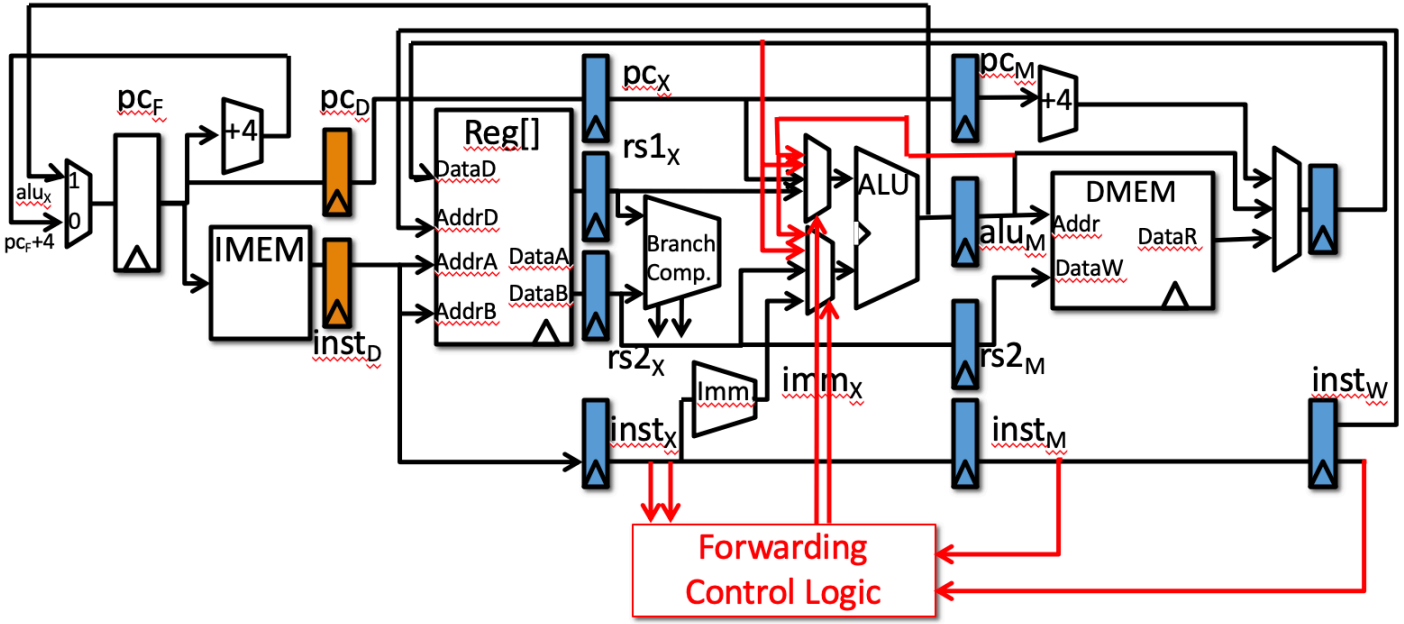


LW **t1**, 0(s1)
SW t2, 0(**t1**)

ראשית, נשים לב ש LW כותב ל- t1, ו- SW קורא ממנו.

שנית, נדע את ערך t1 רק לאחר שלב ה- Mem, בזמן שפקודת ה- SW תהיה בשלב ה- Mem. אין לנו Forwarding משלב ה- WB לשלב ה- Mem ולכן נצטרך להוסיף בשלב הראשון פקודת NOP אחת.

בשלב הבא, יש לנו Forwarding משלב ה- WB לשלב ה- Execute, ספציפית לתוך הקלט העליון של ה- ALU, ולא נצטרך להוסיף עוד NOPs.



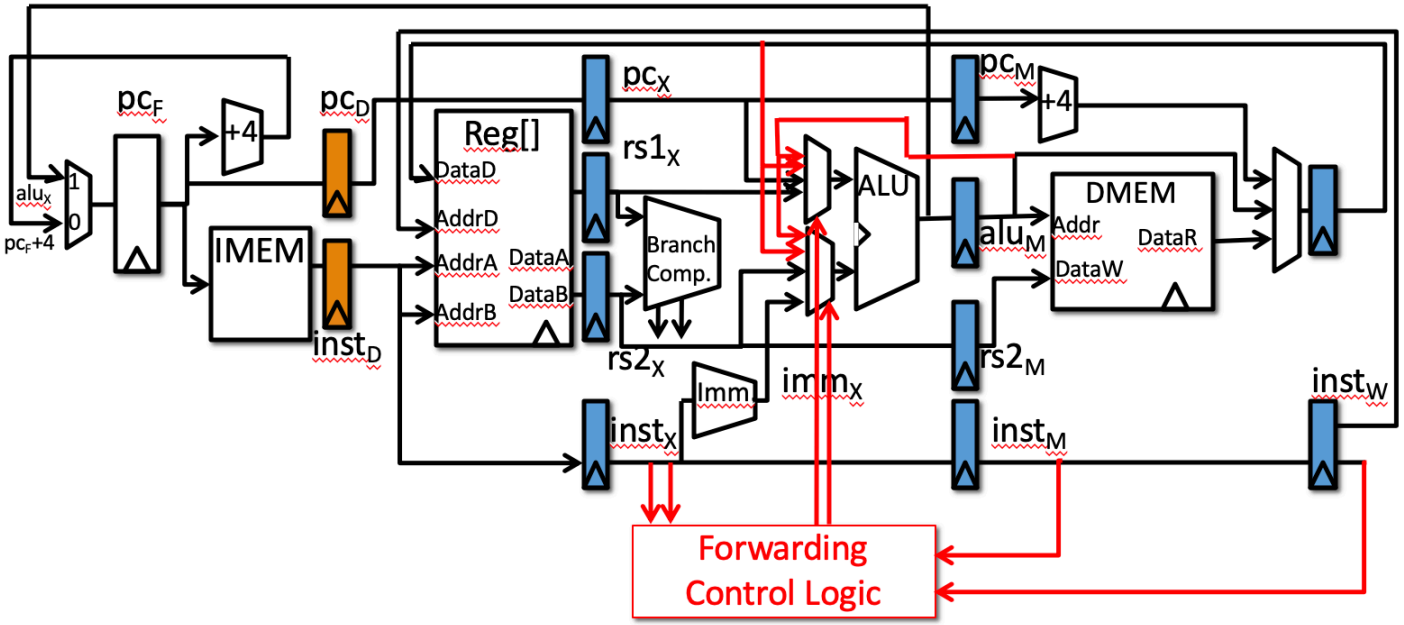
סה"כ נצטרך פקודת NOP אחת

שאלה 3ב – חורף תשע"ט מועד ב' (גרסה מתוקנת ומורחבת)

```
lw t1, 0(s1)
sw t1, 0(s0)
```

מה המספר המינימלי של פקודות *NOP* שיש להוסיף לקוד לעיל כדי שירוץ כשורה?

א- 0
ב- 1
ג- 2
ד- 3
ה- 4

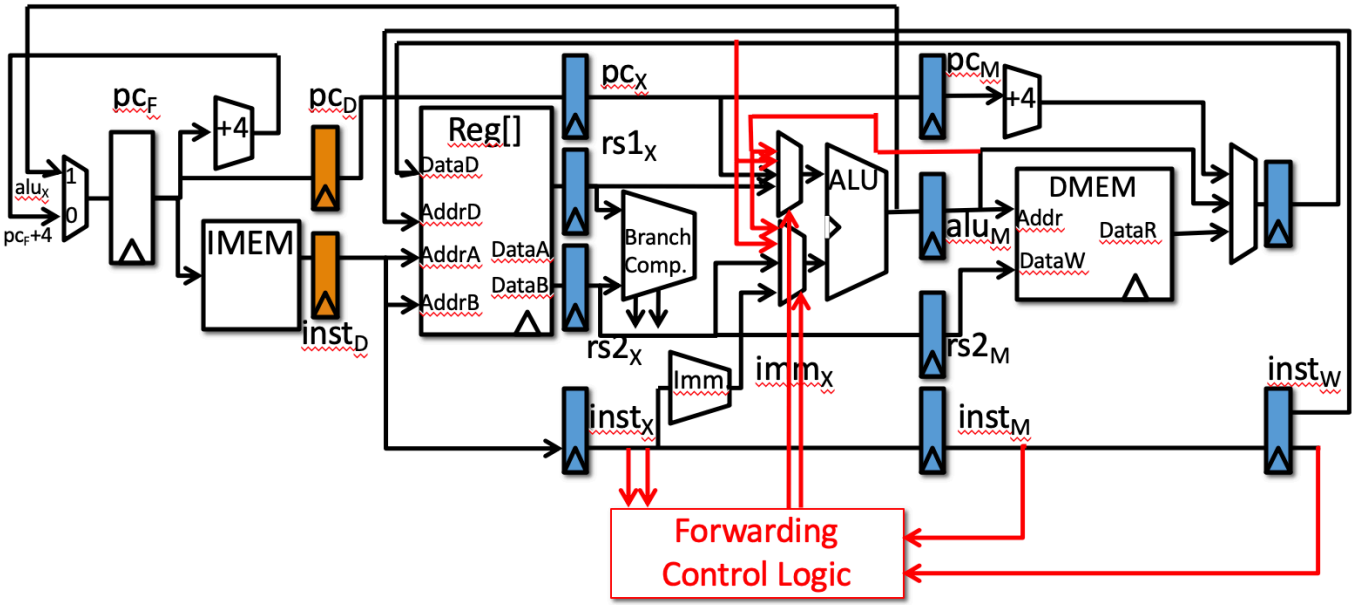


```
lw t1, 0(s1)
sw t1, 0(s0)
```

ראשית, נשים לב ש LW כותב ל- t1, ו- SW קורא ממנו.

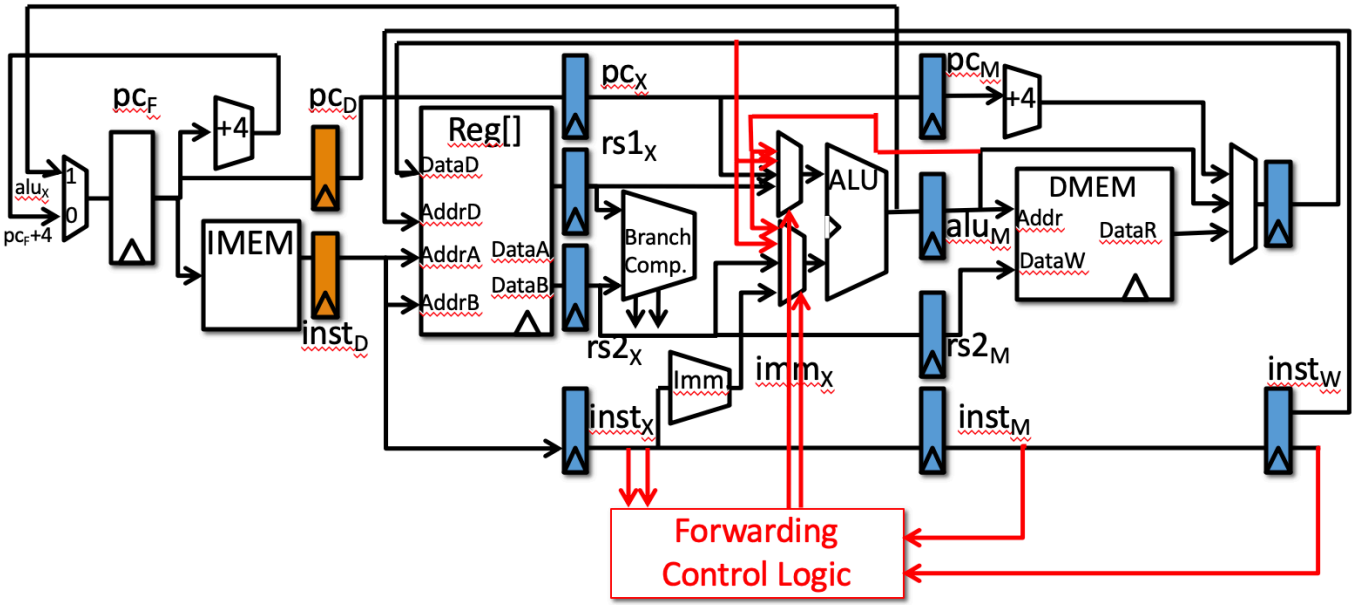
שנית, נדע את ערך t1 רק לאחר שלב ה- Mem, בזמן שפקודת ה- SW תהיה בשלב ה- Mem. אין לנו Forwarding משלב ה- WB לשלב ה- Mem ולכן נצטרך להוסיף בשלב הראשון פקודת NOP אחת.

כעת נשים לב שיחידת ה- Forwarding כלל לא רלוונטית במקרה זה, משום שהערך שמועבר בין שלב ה- WB לשלב ה- EXE מתבצע הוא ערך ה- Addr לזיכרון, ולא ה- Data ולכן נצטרך להוסיף NOP נוסף.



```
lw t1, 0(s1)
sw t1, 0(s0)
```

לבסוף נתון שה- RegFile לא מכיל Forwarding, ולכן נצטרך להוסיף NOP שלישי.
סה"כ נפריד לחלוטין בין שתי הפקודות ע"י 3 פקודות NOP.



שאלה 3ג – חורף תשע"ט מועד ב' (גרסה מתוקנת ומורחבת)

```
lw t1, 0(s1)
sw t1, 0(s0)
```

סעיף ג' (לא הופיע במבחן)

הצע תיקון ל- Datapath ע"י הוספת / שינוי חוטים ובוררים על מנת להפחית את כמות פקודות ה- NOP הנדרשות. כמה פקודות NOP נדרשות כעת?

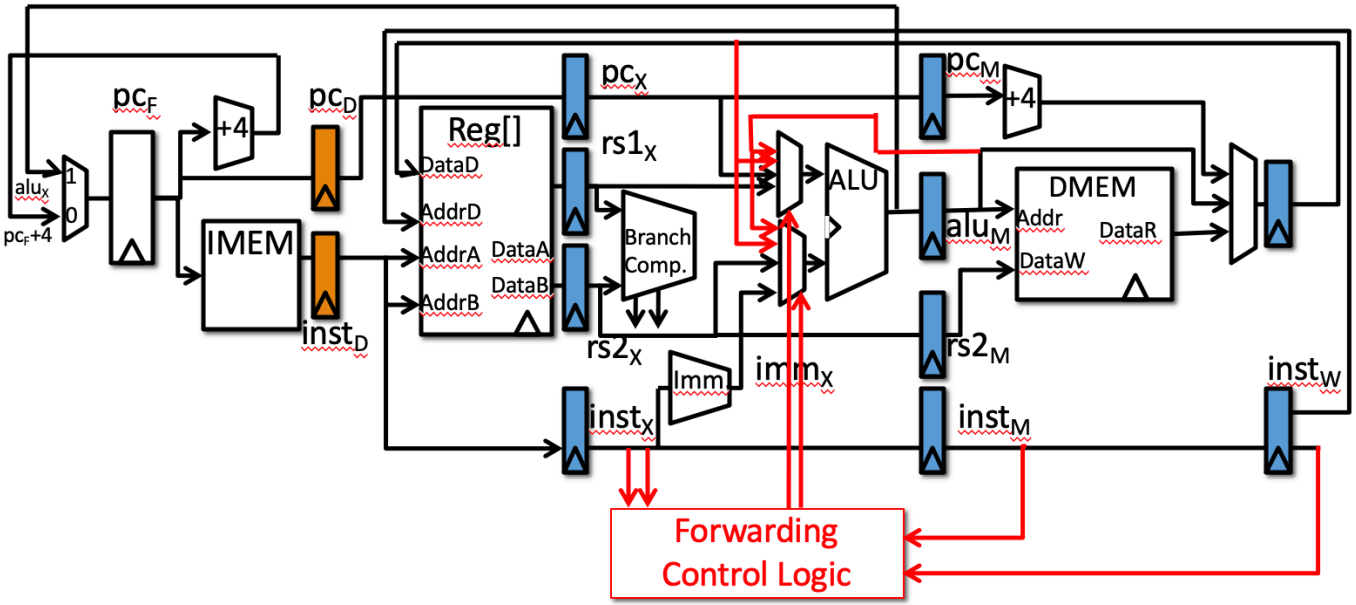
א- 0

ב- 1

ג- 2

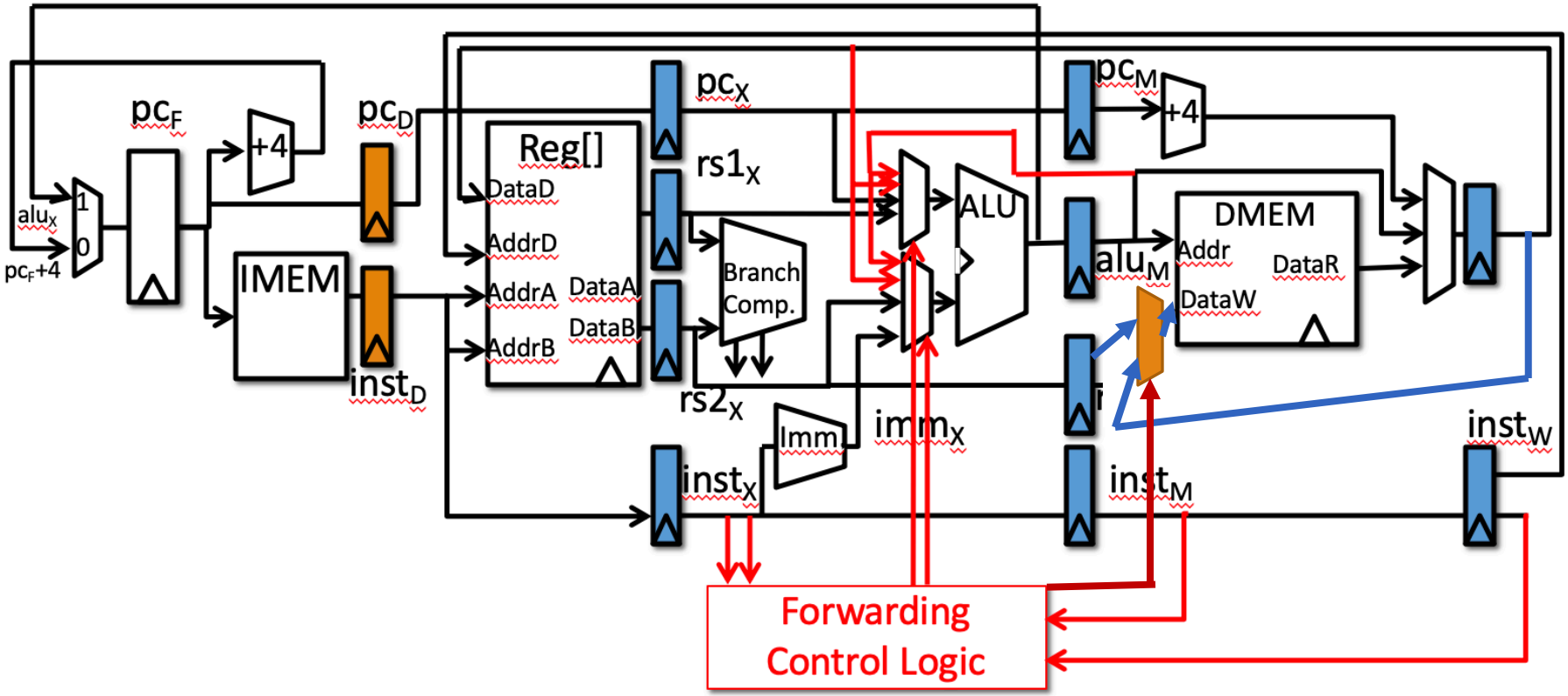
ד- 3

ה- 4



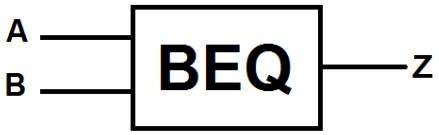
```
lw t1, 0(s1)
sw t1, 0(s0)
```

ע"י הוספת Forwarding משלב ה- WB לשלב ה- MEM (לתוך כניסת ה- DATA של ה- DMEM)



שאלה 10 – מימוש מערכת צירופית (5 נקודות)

מעוניינים לממש **רכיב צירופי** *BEQ* בעל שתי כניסות, כל אחת ברוחב של 8 סיביות, ויציאה אחת ברוחב של סיבית אחת.



הרכיב מממש את הפונקציה הבאה:

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

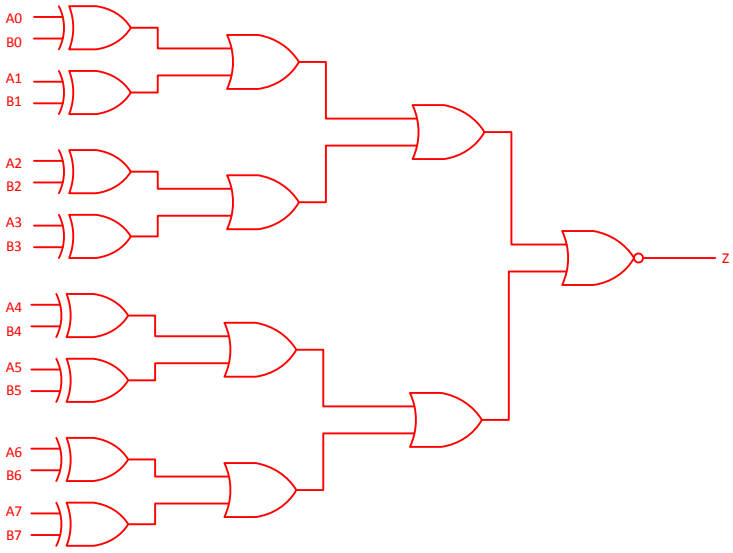
לשם כך, ברשותכם כמות אינסופית של השערים הלוגיים:
$\{AND, OR, NOT, XOR, NAND, NOR\}$ בעלי שתי כניסות כל אחד (למהפך יש כניסה אחת).
זמן ההשהיה של כל שער הוא $t_d = 1\ ns$.

שימו לב בשאלה זו אנו מעוניינים למצוא מימוש בעל זמן השהיה מינימלי **ולא** מספר שערים מינימלי. במילים אחרות, עדיפות ראשונה זמן השהיה מינימלי, עדיפות משנית כמות שערים מינימלית.

מבין התשובות הבאות, מהו זמן ההשהיה המינימלי האפשרי של הרכיב *BEQ* ?

א- 3
ב- 4
ג- 5
ד- 6
ה- תשובות א'-ד' לא נכונות

פתרון: תשובה ב



שאלה 11 – SC Vs. MC (15 נקודות)

נתונים הזמנים הבאים עבור השלבים השונים במעבד *RISC-V*:

IF	*ID*	*EXE*	*MEM*	*WB*
200ps	*100ps*	*200ps*	*200ps*	*100ps*

בחברת *"all the single processors"*, מתמחים בייצור מעבדי *Single Cycle RISCV*.
מולם יש חברה מתחרה, *"Multi Python"*, המייצרת מעבדי *Multi Cycle RISCV*.

נתונה תכנית המורכבת אך ורק מפקודות *lw* ו- *branch*.

נתון שהיחס בין כמות הפקודות מהסוגים השונים הוא: $\frac{\#lw}{\#branch} = a$

א- (5 נקודות) מהם ערכי *a* עבורם המעבד מסוג *Single Cycle* יסיים את התכנית לפני המעבד מסוג *Multi Cycle*? הסבירו.

זמן המחזור של הSingle Cycle הוא T = 800ps.
זמן המחזור של הMulti Cycle הוא T = 200ps.
פקודת lw לוקחת 5 מחזורים בMulti Cycle כלומר 1000ps.
פקודת branch לוקחת 3 מחזורים בMulti Cycle כלומר 600ps.

נניח שבתכנית יש x פקודות מסוג lw ו-y פקודות מסוג branch. כלומר x/y = a
לכן החישוב הוא:

$$800(x+y) < 1000x + 600y$$
$$200y < 200x \quad // : 200y$$

$$a > 1$$

עבור הסעיפים ב', ג' נתון ש a = 0.5.

אחד המהנדסים בחברת "all the single processors" מצא דרך לייעל את הארכיטקטורה כך ששלב ה- EXE בלבד יקח פחות זמן. כלומר שהזמן שלוקח שלב ה- EXE הוא $200 - \delta$.

ב- (5 נקודות) בהנחה שהשיפור אינו מיושם במעבד מסוג *Multi Cycle*, מהם ערכי δ עבורם המעבד מסוג *Single Cycle* יסיים את התכנית לפני המעבד מסוג *Multi Cycle*?

זמן המחזור של הSingle Cycle הוא $T = 800 - \delta\ ps$.
זמן המחזור של הMulti Cycle הוא T = 200ps.
פקודת lw לוקחת 5 מחזורים בMulti Cycle כלומר 1000ps.
פקודת branch לוקחת 3 מחזורים בMulti Cycle כלומר 600ps.
לכן החישוב הוא:

$$(800 - \delta)(x + y) < 1000x + 600y$$

$$200y < 200x + \delta(x + y) // : y$$

$$200 < 200a + \delta(1 + a)$$

$$100 < 1.5\delta$$

$$\frac{200}{3} < \delta$$

בגלל ריגול תעשייתי, כעת גם במעבדי ה-*Multi Cycle*, אימצו את השיפור של שלב ה-*EXE*.

ג- (5 נקודות) בהנחה שהשיפור מיושם בשני המעבדים, מהם ערכי δ עבורם המעבד מסוג Single Cycle יסיים את התכנית לפני המעבד מסוג Multi Cycle?

זמן המחזור של Multi Cycle לא השתנה כי יש שלבים אחרים ארוכים יותר (כמו שלב הIF).
לכן התשובה תהיה זהה לסעיף ב'.

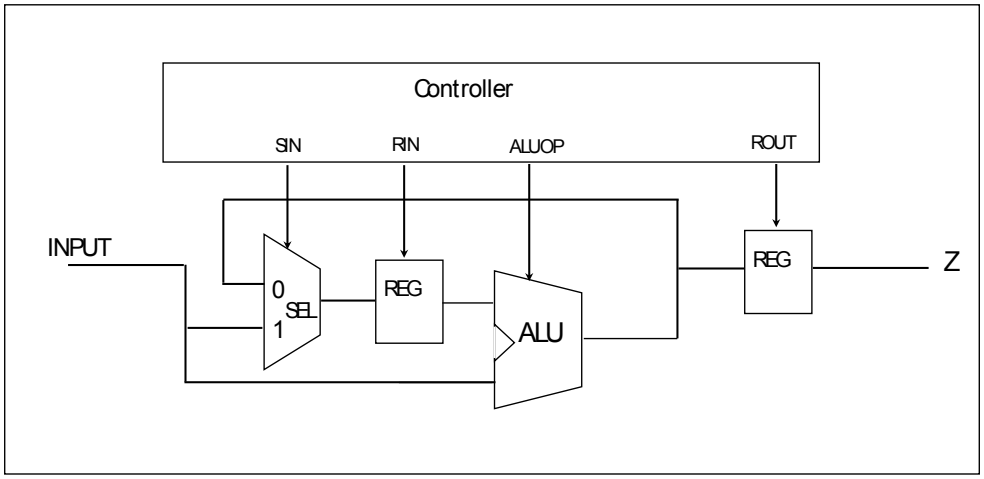
שאלה 12 – Controller-Datapath (15 נקודות)

נתונה מערכת Controller-Datapath. למערכת כניסה בעלת 8 סיביות ויציאה בעלת 8 סיביות. המערכת מקבלת מספר x בעל 8 סיביות, במחזור הבא היא מקבלת מספר y בעל 8 סיביות, ובמחזור הבא היא מקבלת מספר w בעל 8 סיביות.
לבסוף המערכת מוציאה ערך שלם שהוא תוצאת הפעולה $z = \frac{x+y}{w}$, כלומר ערך שלם תחתון לתוצאת החילוק בין (x+y) ל- w. התהליך חוזר על עצמו שוב ושוב.
כלומר, בכניסת המערכת יופיעו המספרים (משמאל לימין) ... $x_1, y_1, w_1, x_2, y_2, w_2,$.
מוצא המערכת מתעדכן לאחר קבלת 3 המספרים x, y, w, ונשאר קבוע למשך 3 מחזורים.

לדוגמה:

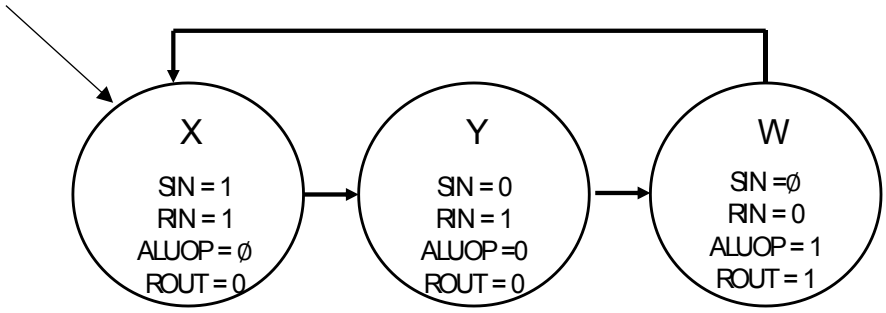
# Cycle	1	2	3	4	5	6	7	8	9
Input	x_1	y_1	w_1	x_2	y_2	w_2	x_3	y_3	w_3
	10	2	3	5	5	2	13	17	3
Output	xxx	xxx	xxx	4	4	4	5	5	5

ה- Data path נתון והינו:



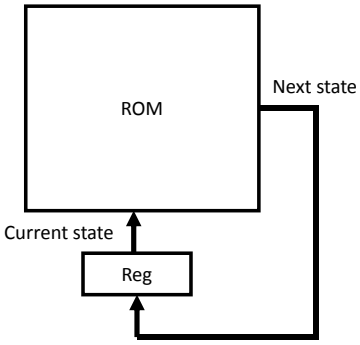
רכיב ה- ALU מסוגל לבצע פעולות חיבור וחילוק בלבד. כאשר ALUOP=0, הוא מבצע פעולת חיבור, וכאשר ALUOP=1 הוא מבצע פעולת חילוק בין הקלט העליון שלו לתחתון (כלומר עליון חלקי תחתון).

א- (5 נקודות) בנו את מכונת המצבים מסוג **Moore** שתתאר את ה- Controller. עליכם לציין בכל מצב מה יהיו הערכים של כל סיביות הבקרה במהלך אותו מצב. סיביות בקרה אלו הן יציאות ה- Controller. **<u>על מכונת המצבים להיות בעלת מספר מינימלי של מצבים.</u>**



ב- (5 נקודות) הוחלט על מימוש ה- Controller באמצעות ROM ורגיסטר מצב נוכחי **בלבד.**

להזכירכם, מערכת כזאת נראית כך:



מה קיבולת ה-ROM המינימלית (מספר סיביות נתונים) לצורך המימוש? הסבירו.

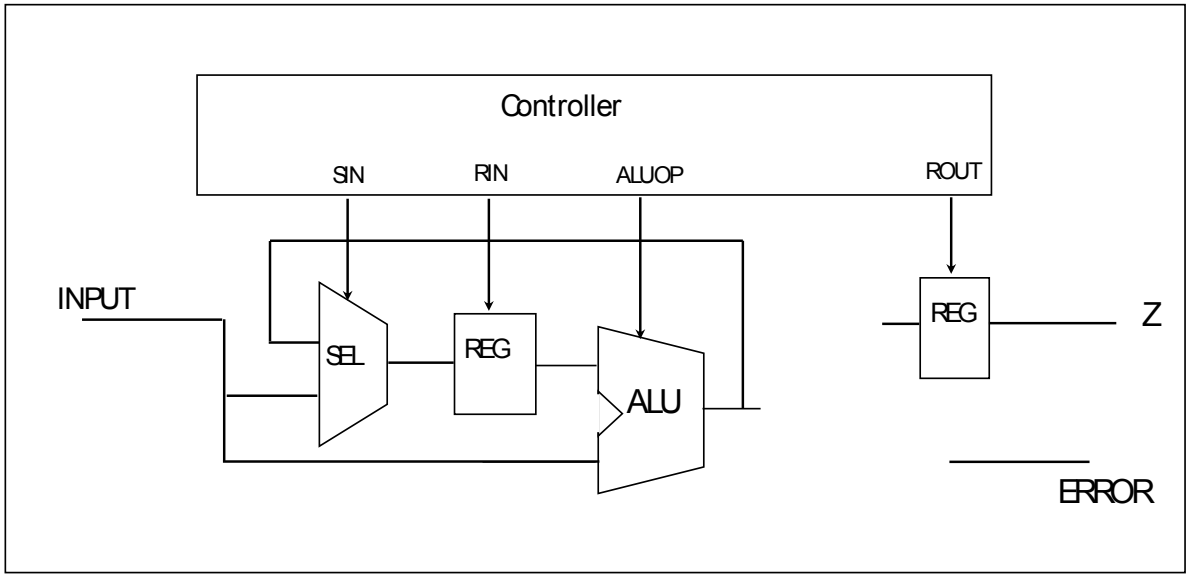
מספר המצבים הנדרש הוא 3, ולכן על מנת לייצג את המצב נידרש ל-2 ביטים. בנוסף ישנן 4 סיביות בקרה, ולכן נידרש ל- $ROM_{size} = 2^2(2+4) = 24bit$.

קיבלנו גם את התשובה לפיה מספר השורות בטבלת הרום לא חייב להיות חזקת 2.
כלומר- $ROM_{SIZE} = 3(2+4) = 18bit$

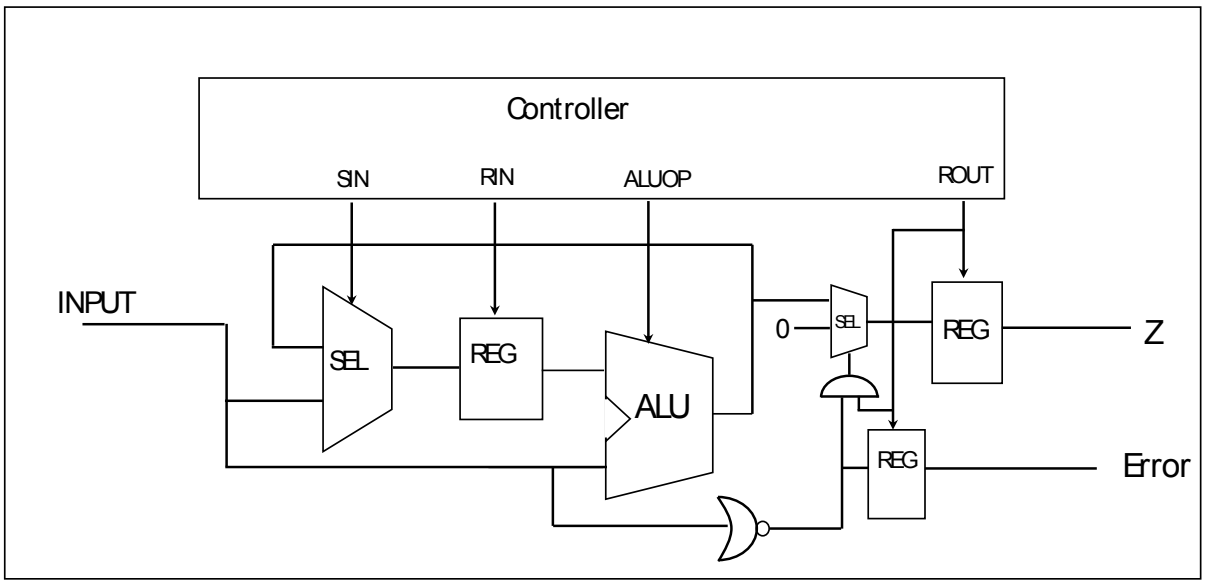
ג- (5 נקודות) הסטודנט שמימש את המערכת שם לב שהיא לא מטפלת במצב שבו w=0 (כלומר, במצב בו מחלקים ב- 0). הסטודנט רוצה לשנות את המערכת כך שבהינתן w=0, המערכת תוציא בתום החישוב את המוצאים error=1 וגם z=0 (במקום להוציא את תוצאת החילוק).

בתרשים שלהלן, הוסיפו את הלוגיקה הנדרשת ליישום השינוי המבוקש.

ב -Datapath מותר להשתמש בשערים לוגיים, רגיסטרים, סלקטורים, חוטים וקבועים ללא הגבלה. ב- Controller באפשרותכם לשנות או להוסיף מוצאים למכונת המצבים, אך **אינכם רשאים להוסיף מצבים למכונת המצבים**.



אחד הפתרונות האפשריים:



שאלה 13 - ALU (5 נקודות)

נתון מעבד מסוג Single-Cycle ומעבד מסוג Multi-Cycle.
בדיקה העלתה שבשניהם יש תקלה: כאשר ALUsel = sub, רכיב ה- ALU מבצע דווקא add.
בהינתן תקלה זו ובהתיחס בנפרד לכל אחד משני המעבדים, הקף בעיגול את הפקודות שירוצו כשורה (כלומר לא יושפעו מהתקלה).

Multi Cycle	Single Cycle
Every R-type instruction	Every R-type instruction
LW	LW
SW	SW
BEQ	BEQ
JAL	JAL
אף פקודה לא תרוץ כשורה	אף פקודה לא תרוץ כשורה

פתרון:

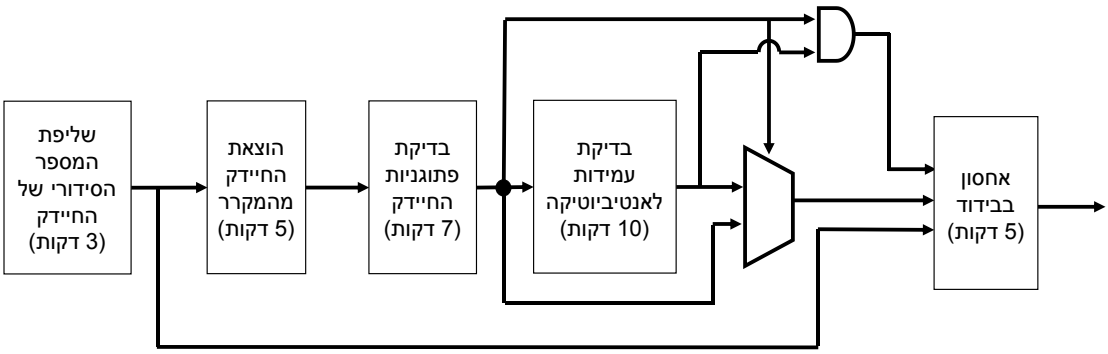
עבור שני המעבדים לא ניתן לבצע את כל פקודות ה- R-type.

פקודות LW ו- SW, מבצעות פעולת Add ולכן יעבדו כשורה על שני המעבדים. בפקודת BEQ ה- ALU צריך לחשב את PC=PC+imm ב- Single Cycle ולכן יעבוד, אך ב- Multi-Cycle יש צורך לחשב גם את ההשוואה בין הרגיסטרים ע"י חיסור, ולכן הפקודה לא תוכל לרוץ כשורה.

פקודת JAL תרוץ כהלכה בשניהם, כי יש צורך בחישוב PC=PC+immediate.

שאלה 14 (15 נקודות)

מהנדסת ביוטכנולוגיה מעוניינת לשפר את תהליך בדיקת החיידקים במעבדה.
לצורך בניית המערכת, המהנדסת פנתה לשותפתה למעונות, שעברה את הקורס "מערכות ספרתיות ומבנה המחשב", שהציעה את המערכת הבאה:



זמני ההשהיה של המערכות רשומים על גבי השרטוט לעיל כך שהזמן שרשום על גבי כל מערכת מהווה גם את זמן ה- t_{pd} וגם את זמן ה- t_{cd} שלה. זמני ההשהיה של שאר הרכיבים הלוגיים זניחים. אופן פעולת המערכת יוסבר בכל שאלה.
בנוסף, נתון שהמערכת מקיימת את המשטר הסטטי לפיו אין לשנות את הכניסה לפני שמוצא המעגל סיים להתעדכן בוודאות.

א- (5 נקודות) המהנדסת מעוניינת להפעיל את המכשיר במשך הלילה כדי לבדוק את חיידקי המעבדה. לשם כך היא צריכה להגדיר זמן מחזור למערכת **הצירופית**, כך שבכל זמן מחזור יישלף מספר סידורי של חיידק כלשהו. מה מספר הבדיקות המרבי שיכולה המערכת לבצע במשך 10 שעות? הסבירו.

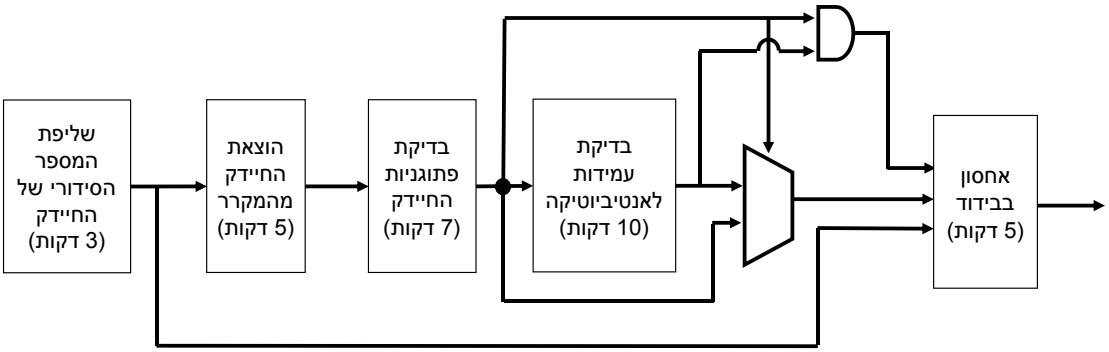
בכל 30 דקות ניתן יהיה לבצע חישוב חדש במערכת. כדי לחשב זאת, יש לבחון את המסלול הארוך ביותר בין כניסה ליציאה.
ולכן ב-10 שעות ניתן לבצע $\frac{10}{0.5} = 20$ בדיקות

ב- (10 נקודות) מוצע לצנר את המערכת כדי להגדיל את תפוקתה בעדיפות ראשונה, ושימוש במינימום רגיסטרים בעדיפות שנייה. המהנדסת שמה לב שיש במלאי רק רגיסטרים עם הנתונים הבאים (בדקות):

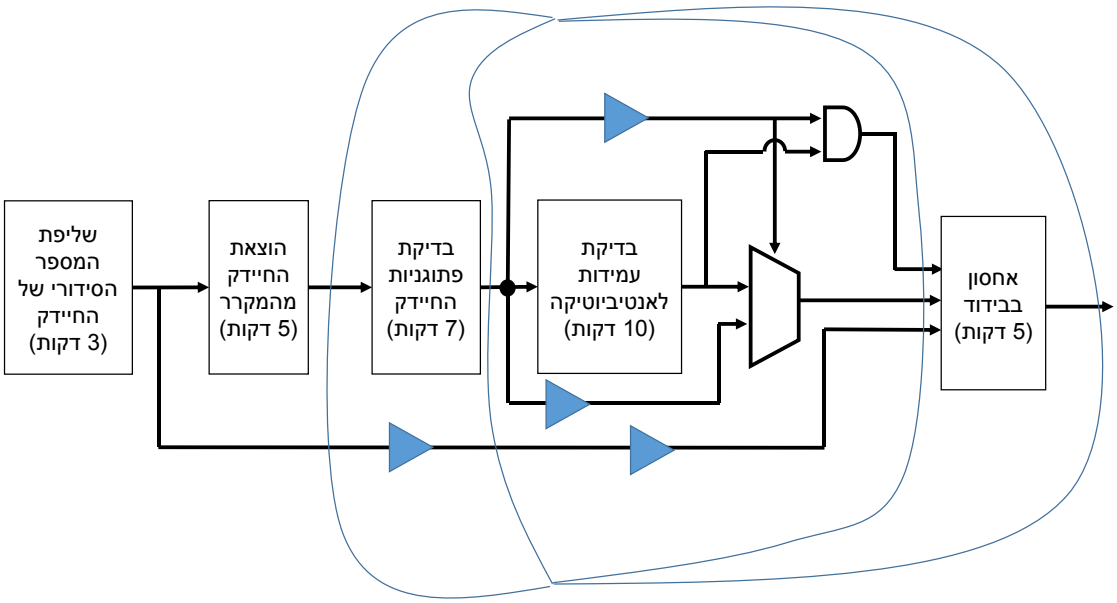
$t_{cCQ} = 0.5min$
$t_{pCQ} = 2min$
$t_{hold} = 1min$
$t_{setup} = 0.5min$

בנוסף, יש במעבדה מערכות השהייה שלא מבצעות פעולה לוגית כלשהי, אך יש להן את זמן ההשהיה: $t_{pd} = t_{cd} = 0.5min$. בכל מקום שתרצו להוסיף רכיב השהייה סמנו זאת בעזרת משולש. כלומר:

צנרו את המערכת על גבי הציור להלן, כך שיתקבל המעגל **התקין** הנדרש.



פתרון: ניתן לצנר את המערכת באופן הבא:



שימו לב שהצינור לעיל מייצר מצב ששני רגיסטרים מחוברים זה לזה בעזרת חוט בלבד (ללא השהיות). היות וה- t_{cCQ} של הרגיסטרים קטן מה- t_{hold} שלהם, נדרש להוסיף השהייה אחת בין כל רגיסטר לרגיסטר. סה"כ, זמן המחזור נקבע לפי הרכיב עם ההשהייה הארוכה ביותר, בתוספת זמן ההשהייה וה- *setup* של הרגיסטרים:
$T = 10\min + 2min + 0.5min = 12.5min$. וכך נקבל את ה- *Throughput* הדרוש.