v

מערכות ספרתיות ומבנה המחשב (044252)

סמסטר חורף תש"פ

בחינה סופית – מועד ב

פתרון

3 במרץ 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 13 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.
- 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 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 קוד ה-SystemVerilog הבא, כאשר counter הוא module הממש מונה בעל 2 ביטים (סופר מ-0 עד 3):

```
module my_module (
    input logic clk,
    input logic rst,
    input logic a,
    output logic q
);
    typedef enum { S0_st, S1_st, S2_st } sm_type;

    sm_type current_state;
    sm_type next_state;

    always_ff @(posedge clk, posedge rst) begin
        if (rst == 1'b1) begin
            current_state <= S0_st;
        end
        else begin
            current_state <= next_state;
        end
    end

    always_comb begin
        case (current_state)
            S0_st: begin
                next_state = S1_st;
                q = 1'b0;
            end
            S1_st: begin
                next_state = S2_st;
                q = 1'b0;
            end
            S2_st: begin
                next_state = S0_st;
                q = a;
            end
            default: begin
                next_state = S0_st;
                q = 1'b0;
            end
        endcase
    end

endmodule
```

```
module my_module2 (
    input logic clk,
    input logic rst,
    output logic out
);
    logic [1:0] cnt;
    counter cnt_inst(.clk(clk), .rst(rst), .cnt(cnt));
    my_module inst (.clk(clk), .rst(rst), .a(cnt[0] & cnt[1]),
.q(out));

endmodule
```

```
module counter (
    input logic  clk,
    input logic  rst,
    output logic [1:0] cnt
);
    always_ff @(posedge clk, posedge rst)
    begin
        if (rst == 1'b1) begin
            cnt <= 2'b00;
        end
        else begin
            cnt <= cnt + 1;
        end
    end
endmodule
```

Duty cycle מוגדר עבור אות מחזורי בתור החלק היחסי מתוך זמן מחזור האות שבו האות בעל ערך '1' (למשל, duty cycle של אות שעון סטנדרטי הוא 1/2). מהו ה-duty cycle של הסיגנל out (היציאה של my_module2)?

א. 1/12
ב. 1/6
ג. 1/4
ד. 1/3
ה. 1/2

פתרון

התשובה הנכונה היא תשובה א'.

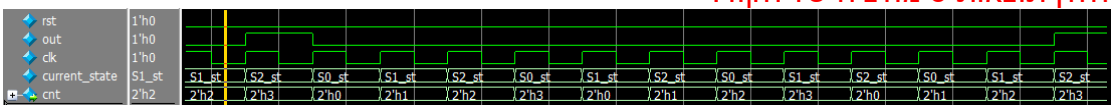
הקוד של my_module דומה לקוד של Clock Divider שראינו בסדנה, עם השינוי הבא:

```
module my_module (
    input logic clk,
    input logic rst,
    input logic a,
    output logic q
);
    typedef enum { S0_st, S1_st, S2_st } sm_type;

    sm_type current_state;
    sm_type next_state;

    always_ff @(posedge clk, posedge rst) begin
        if (rst == 1'b1) begin
            current_state <= S0_st;
        end
        else begin
            current_state <= next_state;
        end
    end

    always_comb begin
        case (current_state)
            S0_st: begin
                next_state = S1_st;
                q = 1'b0;
            end
            S1_st: begin
```

```
                next_state = S2_st;
                q = 1'b0;
            end
            S2_st: begin
                next_state = S0_st;
                q = a;
            end
            default: begin
                next_state = S0_st;
                q = 1'b0;
            end
        endcase
    end

endmodule
```

ו.

כלומר, כדי שהיציאה תהיה '1', ה-FSM צריך להיות במצב S2_st וגם a צריך להיות '1'. ה-FSM מגיע למצב S2_st כל שלושה מחזורי שעון ותוצאת ה-AND של ביטי ה-counter היא '1' כל ארבעה מחזורי שעון (כאשר ה-counter בעל ערך '11') ולכן היציאה תהיה '1' בכל מכפלה משותפת של 3 ו-4, כלומר כל 12 מחזורי שעון.
להלן תוצאות סימולציה של הקוד:



שאלה 2 (5 נקודות)

עבור פקודות האסמבלי הבאות, איזו פקודה **לא** ניתן לממש כפקודה אמיתית במעבד הmulti-cycle RISC-V (להזכירכם פקודה אסמבלי אמיתית רצה כפקודה אחת על המעבד)?

ניתן לבצע שינויים בבקר והוספת muxes וחיווטים במסלול הנתונים של המעבד ולהתאים את זמן המחזור, אך **אסור** לבצע שינויים ביחידות Register file, Memory, ALU.

א. פקודת swap rd, rs אשר מחליפה בין תוכנם של שני הרגיסטרים.

ב. פקודת addi32 rd, rs, imm אשר מוסיפה ערך immediate בגודל 32 ביט לערך השמור ברגיסטר rs ושומרת ב-rd.

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

ד. פקודת sub3 rd, rs1, rs2 אשר מבצעת את הפעולה reg[rd]=reg[rd]-reg[rs1]-reg[rs2] (שומרת את התוצאה לרגיסטר rd).

ה. ניתן לממש את כל הפקודות הנ"ל.

פתרון

תשובה ב' נכונה.

לא ניתן לממש add32i מכיוון שלא ניתן לקודד בפקודת אסמבלי (בגודל 32 ביט) גם opcode, גם מספר רגיסטר, וגם ערך imm של 32 ביטים.

שאלה 3 (5 נקודות)

משדר ומקלט מתקשרים באמצעות קו יחיד לפי השיטה הנלמדת בקורס.
נתון כי בעת השידור ישנו רעש אשר הופך באופן אקראי סיבית אחת (מתוך 8 סיביות המידע) בכל שידור ('0' לוגי הופך ל-'1' ולהפך). סיביות start ו stop נשארות תקינות.
מהו השינוי שיוכל להקטין בצורה המשמעותית ביותר את הסיכוי לשגיאה בשחזור המידע במקלט?

א. שימוש בסיבית זוגיות אחת לכל 8 סיביות המידע.
ב. שימוש בסיבית זוגיות אחת לארבעת סיביות המידע הראשונות וסיבית זוגית לארבעת הסיביות הבאות.
ג. שידור כל סיבית פעמיים.
ד. שידור כל סיבית שלוש פעמים.
ה. לא ניתן לשחזר את המידע בכלל.

פתרון

ד'
סעיפים א-ג מאפשרים לזהות שגיאה אך לא לתקן אותה.
שידור כל סיבית 3 פעמים מאפשר לזהות את השגיאה וניתן לתקן אותה כאשר קובעים את הערך לפי הערך של רוב הסיביות מבין שלושת הסיביות.

שאלה 4 (5 נקודות)

ממשו את הפונקציה F(x,y,z)=xy+z' בעזרת שערי NAND עם 2 כניסות בלבד.
מהו מספר שערי ה-nand המינימלי שיש צורך להשתמש בהם?

א. 1
ב. 2
ג. 3
ד. 4
ה. אין אפשרות לממש את הפונקציה הזו באמצעות שימוש בשערי nand

פתרון

ב'
xy+z'=((xy)'z)'=((x nand y) nand z)

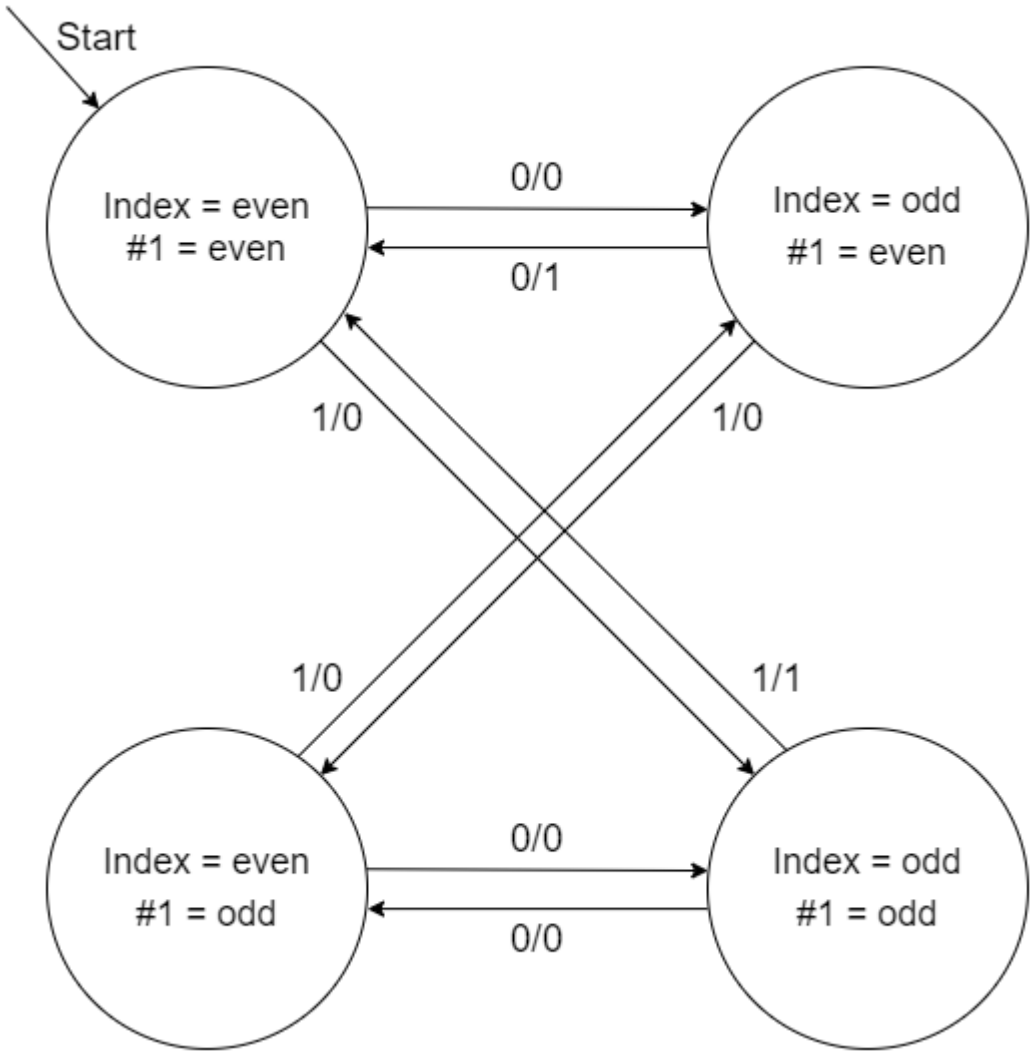
שאלה 5 (5 נקודות)

נדרש לתכנן מערכת עקיבה מסוג מילי בעלת כניסה יחידה ויציאה יחידה המפיקה Z=1 אמ"מ אינדקס הכניסה עד כה הוא זוגי ומספר האחדים עד כה הוא זוגי. יש להניח כי המערכת מתחילה ממצב של רצף אפסים באורך זוגי. כמה מצבים יהיו במכונה המצומצמת?

א. 3
ב. 4
ג. 5
ד. 6
ה. 8

פתרון

ב'
נשרטט מכונה אשר פעולת כנדרש:

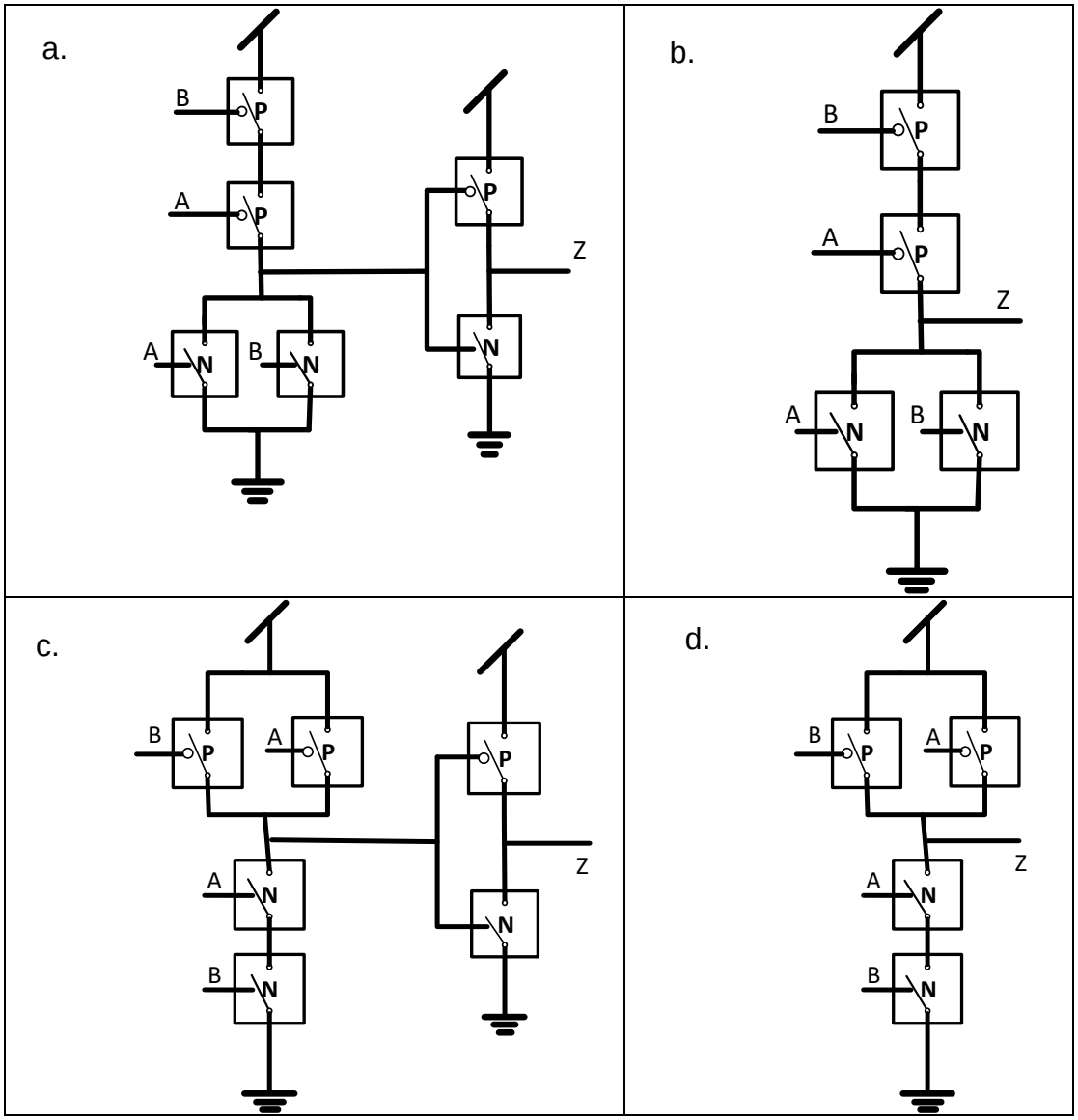


כלומר דרושים 4 מצבים לצורך מימוש המכונה.

שאלה 6 (5 נקודות)

נתונים מספר שערים לוגים אשר מומשו על ידי מתגים.
מבין האפשרויות הבאות, מהו תכנון המתגים אשר מממש שער or $(Z = A + B)$?

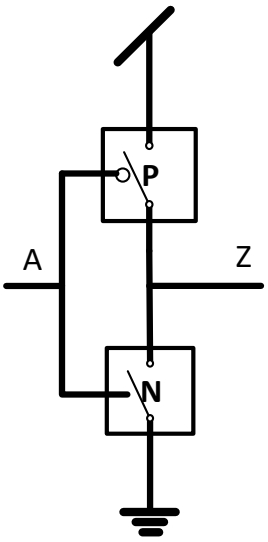
תזכורת: מתג P מחובר כאשר כניסתו היא 0, ומתג N מחובר כאשר כניסתו היא 1.



א. איור a
ב. איור b
ג. איור c
ד. איור d
ה. אף תשובה לא נכונה

פתרון

התשובה הנכונה היא תשובה א'.
תשובה ב' – מימוש של שער nor.
תשובה ד' – מימוש של שער $nand$.
שער not במימוש מבוסס מתגים יראה כך:



תשובה א' – זהו בעצם חיבור של מוצא תשובה ה' אל כניסת מהפך, כלומר זהו חיבור של שער nor אל כניסת שער not, לכן התקבל שער or. זו התשובה הנכונה.
תשובה ג' – זהו בעצם חיבור של מוצא תשובה ג' אל כניסת מהפך, כלומר זהו חיבור של שער $nand$ אל כניסת שער not, לכן התקבל שער and.

שאלה 7 (5 נקודות)

כדי לחזק את יכולת גילוי השגיאות של קוד $Gray$ עבור מילה באורך n, כאשר n הוא חזקה שלמה של 2, הוצע להוסיף לכל מילה מקודדת ביט זוגיות אשר ימוקם בביט ה-LSB. לדוגמא, עבור המספר הבינארי 10011, הקידוד אשר יתקבל לאחר המרה לקוד $Gray$ יהיה 11010, ולאחר הוספה של ביט הזוגיות נקבל כי הקידוד הסופי הינו 110101.
הנכם מתבקשים לממש מעגל אשר יבצע את ההמרה מייצוג בינארי אל הייצוג בקוד $Gray$ בתוספת סיביות הזוגיות. לצורך המימוש ניתן להשתמש בשערי xor בעלי **2 כניסות** בלבד.
מהו מספר שערי ה-xor המינימלי (יש למצוא את פונקציית המיתוג המצומצמת ביותר) כתלות ב-n אשר דרוש בכדי שהמעגל יפעל בצורה תקינה?

הערה: כדאי להיעזר בנוסחת ההמרה לקוד $Gray$ המופיעה בדף הנוסחאות.

א. $n-2$
ב. $n-1$
ג. n
ד. $2n-3$
ה. $3n-4$

פתרון:
התשובה הנכונה היא תשובה ב'.
קוד $Gray$ מקיים:

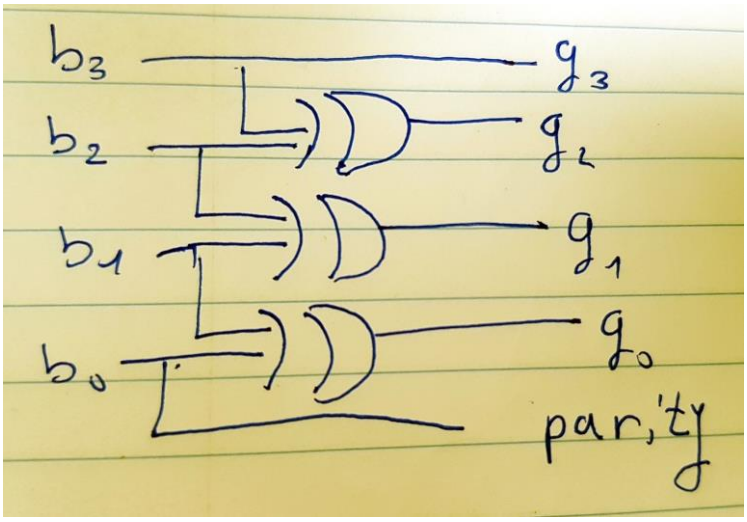
$$g_i = \begin{cases} g_n = b_n \\ g_i = b_i \oplus b_{i+1} \end{cases}$$

אנו יודעים כי בכדי לחשב את ביט הזוגיות של מילה כלשהי, ניתן לבצע פעולות xor בין כל הביטים אשר מרכיבים את המילה. עבור מילה אשר מקודדת בקוד $Gray$ מתקיים:

$$parity\ bit = g_n \oplus g_{n-1} \oplus g_{n-2} \oplus \dots \oplus g_2 \oplus g_1 \oplus g_0 =$$
$$= b_n \oplus (b_n \oplus b_{n-1}) \oplus (b_{n-1} \oplus b_{n-2}) \dots \oplus (b_3 \oplus b_2) \oplus (b_2 \oplus b_1) \oplus (b_1 \oplus b_0) =$$

$= b_n \oplus b_n \oplus b_{n-1} \oplus b_{n-1} \oplus b_{n-2} \dots \oplus b_3 \oplus b_2 \oplus b_2 \oplus b_1 \oplus b_1 \oplus b_0 =$
$= b_0$

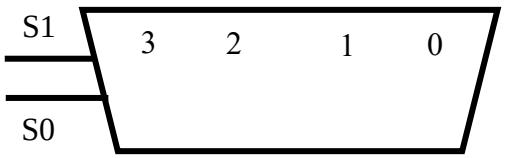
כלומר ביט הזוגיות של המילה בקידוד $Gray$ שווה לביט ה-LSB של המילה בקידוד הבינארי, לכן נוכל להעבירו ישירות מהמילה המקורית ללא צורך בפעולות עם שערים.
מכיוון ש-$b_n = g_n$, נותר לבצע פעולות xor אשר מחשבות את ההמרה לכל g_i אשר מקיים $i \neq n$. כלומר, עלינו לבצע $n-1$ פעולות xor אשר פועלות על 2 ביטים, פעולה עבור כל g_i.
שערי ה-xor אשר מחשבים את g_i יהיו היחידים אשר נזדקק להם, כלומר דרושים $n-1$ שערי xor. לכן התשובה נכונה היא ב'.
דוגמא עבור $n = 4$:



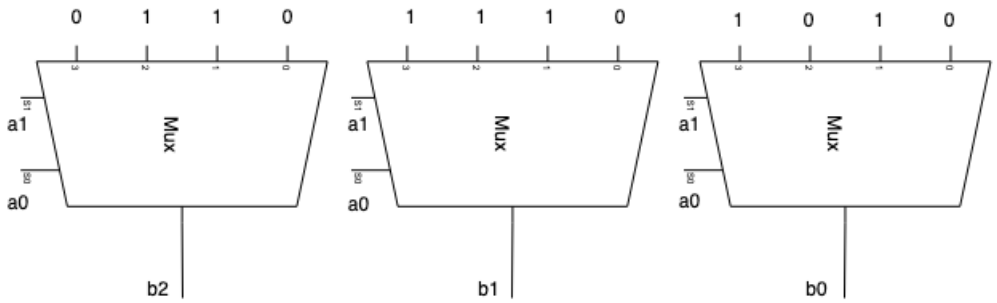
שאלה 8 (5 נקודות)

נתונים מספר מימושים של פונקציה בעזרת בוררים. מבין המימושים המוצגים, אילו מימושים מבצעים המרה של מספר **חיובי** בן 2 סיביות מידע, a_1a_0, המיוצג בייצוג ללא סימן, אל הייצוג שלו כמספר **שלילי** במשלים ל-2 (בעל 3 סיביות) $b_2b_1b_0$? עבור המספר 0 כקלט, הרכיב נדרש להוציא 0 בכל סיביות המוצא.

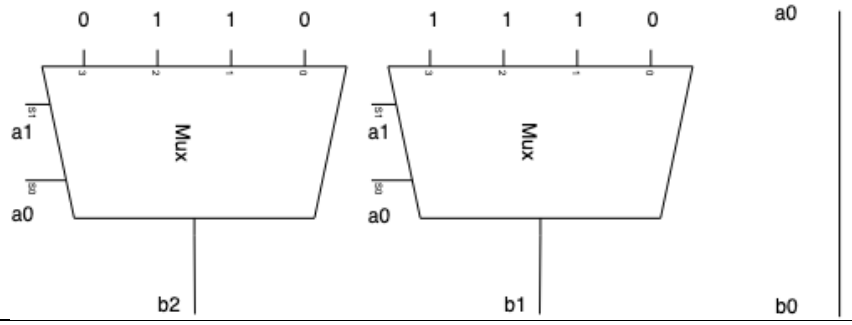
הערה: כל ה-Muxes אשר מופעים מטה, מתוכננים בצורה הבאה:



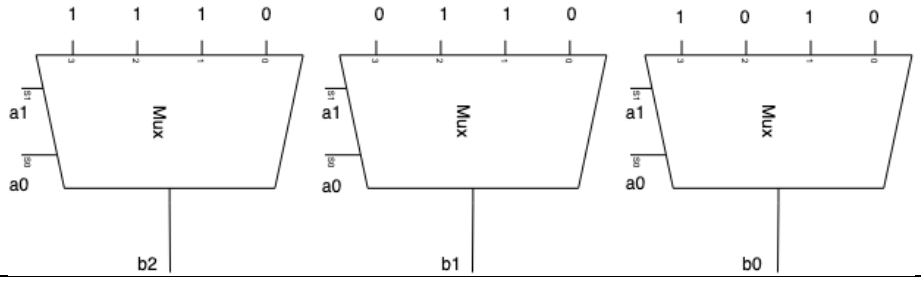
a.


b0
b1
b2
Mux
a0
a1
0 1 0 1
0 1 1 1
0 1 1 0

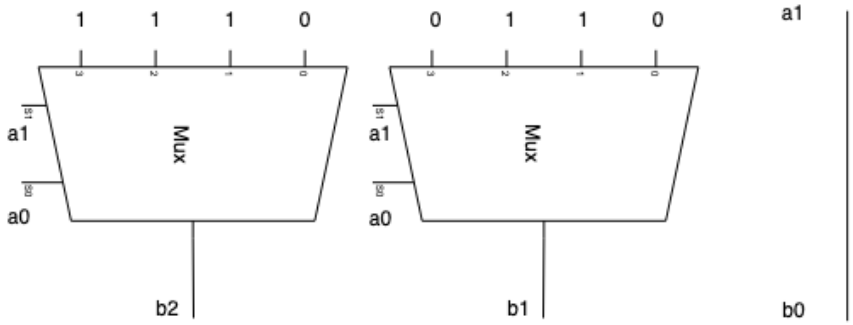
b.


b0
b1
b2
Mux
a0
a1
0 1 1 1
0 1 1 0

c.


b0
b1
b2
Mux
a0
a1
0 1 0 1
0 1 1 0
1 1 1 0

d.


b0
b1
b2
Mux
a0
a1
0 1 1 0
1 1 1 0

א. מימוש a בלבד
ב. מימוש b בלבד
ג. מימוש c בלבד
ד. מימושים a ו-b
ה. מימושים c ו-d

פתרון

ג': שרטוט c בלבד.

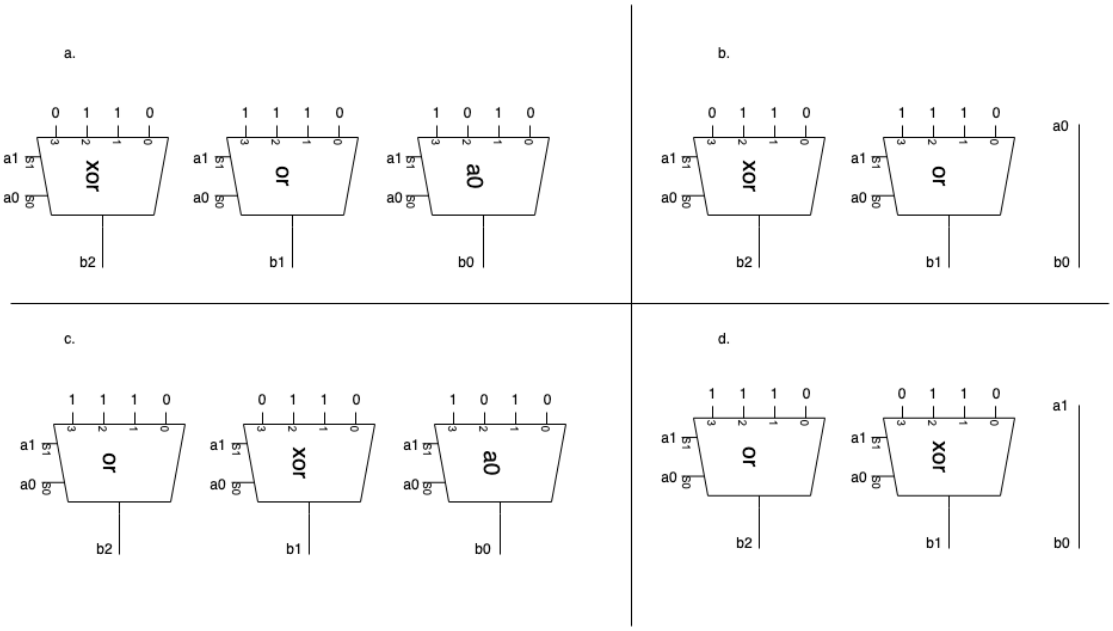
נתבונן בטבלת האמת של ההמרה של מספר חיובי ברוחב 2 ביט אל הייצוג שלו כמספר שלילי (בעל 3 ביט) במשלים ל-2 (נשים לב שבטבלת האמת 0 נשאר ללא שינוי):

a_1	a_0	b_2	b_1	b_0
0	0	0	0	0
0	1	1	1	1
1	0	1	1	0
1	1	1	0	1

מן הטבלה ניתן לראות כי מתקיים:

$$b_2 = a_1 + a_0$$
$$b_1 = a_1 \oplus a_0$$
$$b_0 = a_0$$

עבור השרטוטים הנתונים מתקיים:



לכן רק שרטוט c מממש את הפונקציה המבוקשת.

שאלה 9 (5 נקודות)

נתון קטע הקוד הבא:

```
0x0111 0000      Func:  addi sp, sp, -12
0x0111 0004             addi t1, x0, 2
0x0111 0008             sw ra, 0(sp)
0x0111 000C             sw a0, 4(sp)
0x0111 0010             blt a0, t1, Exit          // Stop condition
0x0111 0014             addi a0, a0, -1
0x0111 0018             jal ra, Func              // Func's output is stored in a0
0x0111 001C             add a1, a0, x0
0x0111 0020             lw a0, 4(sp)
0x0111 0024             sw a1, 8(sp)
0x0111 0028             addi a0, a0, -2
0x0111 002C             jal ra, Func              // Func's output is stored in a0
0x0111 0030             lw a1, 8(sp)
0x0111 0034             add a0, a0, a1
0x0111 0038      Exit:  lw ra 0(sp)               // Return value section
0x0111 004C             addi sp, sp, 12
0x0111 0050             jr ra

0x0112 0000      Main:  addi sp, sp, -4
0x0112 0004             addi a0, x0, 2
0x0112 0008             sw ra, 0(sp)
0x0112 000C             jal ra, Func              // Func's output is stored in a0
0x0112 0010             lw ra, 0(sp)
0x0112 0014             addi sp, sp, 4
0x0112 0018             jr ra
```

התוכנית מתחילה לרוץ מ-Main. מה יהיה ערכו של רגיסטר a0 בסיום ריצת התוכנית?
הערה: ניתן לפתור את השאלה גם ללא שימוש בטבלת מעקב.

א. 1
ב. 2
ג. 3
ד. 4
ה. 0

פתרון

תשובה א'. הפונקציה Func מבצעת חישוב של האיבר ה-n בסדרת פיבונצ'י (להזכירכם, האיבר הראשון הוא עבור n=0) בעזרת קריאות רקורסיביות. עבור הקלט הנתון ($a0 = 2$) ערכו של איבר זה הוא 1.

שאלה 10 (5 נקודות)

נתון מעבד SingleCycle RISCV כפי שנלמד בכיתה התומך בפקודות הנלמדו בכיתה.
נתון שהמעבד עבר שינויים על מנת לתמוך גם בפקודה:

$adMR - add\ memory\ register: adMR\ rd, rs1, rs2$

שמבצעת את הפקודה הבאה:

$Reg[rd] = Mem[Reg[rs1]] + Reg[rs2]$

נתון:

	Timing
Memory Access (Instruction or Data)	1 ns
Read a value from the register file	1 ns
ALU operation	2 ns
Write a value to the register file	1 ns

מהו משך זמן הביצוע המינימלי של הפקודה $addi\ rd, rs1, imm$ אשר נתמכה לפני השינוי?

א. $T = 6ns$
ב. $T = 8ns$
ג. $T = 10ns$
ד. $T = 5ns$
ה. $T = 4ns$

פתרון

עבור הפקודה החדשה נצטרך להשתמש בשלבים הבאים:
fetch, decode, memory, exe, wb

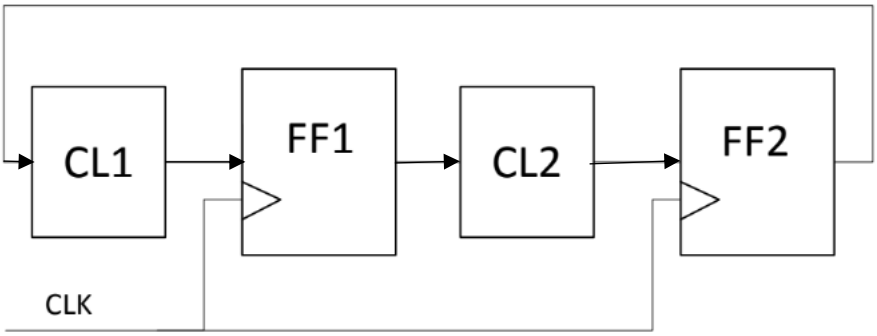
לכן, זמן המחזור יהיה:

$T_{min} = 1 + 1 + 1 + 2 + 1 = 6ns$

(בעצם אין הבדל בזמן, רק בסדר)

שאלה 11 (5 נקודות)

נתון המעגל הבא:



ונתונים גם:

	t_{hold}	t_{setup}	t_{pCQ}/t_{pd}	t_{cd}
FF1	$4ns$	$3ns$	$4ns$	$1ns$
FF2	$4ns$	$4ns$	$7ns$	$2ns$
CL1			$4ns$	$1ns$

מהם נתוני *CL2* שיאפשרו תדר עבודה מקסימלי למעגל:

א. $t_{cd2} \leq 2ns$ $\quad t_{pd2} \leq 6ns$
ב. $t_{cd2} \geq 3ns$ $\quad t_{pd2} \leq 6ns$
ג. $t_{cd2} \geq 3ns$ $\quad t_{pd2} \leq 2ns$
ד. $t_{cd2} \leq 3ns$ $\quad t_{pd2} \leq 4ns$
ה. המעגל לא עומד במשטר הדינאמי.

פתרון

תשובה ה.

תנאי hold עבוד המסלול FF2 אל FF1

$$t_{cd}(FF2) + t_{cd}(CL1) \geq t_{hold}(FF1)$$
$$2ns + 1ns \geq 4ns$$

מי שלא בדק:

מציאת זמן מחזור:

$$T_{min} = 7 + 4 + 3 = 14ns$$

תנאי *setup* עבור *FF2*:

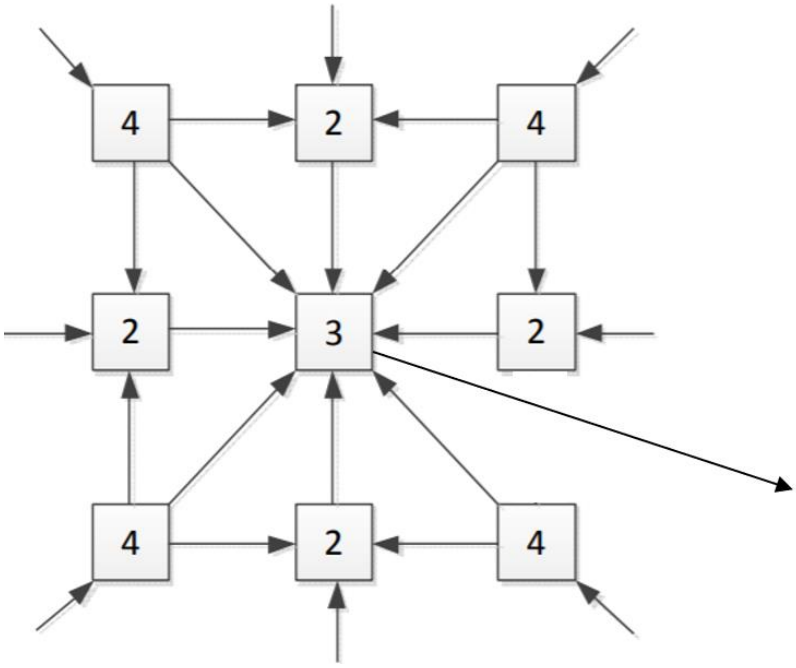
$$4 + t_{pd}(CL2) + 4 \leq 14 \rightarrow t_{pd}(CL2) \leq 6ns$$

תנאי *hold* עבור *FF1*:

$$1 + t_{cd}(CL2) \geq 4 \rightarrow t_{cd}(CL2) \geq 3ns$$

שאלה 12 (5 נקודות)

נתונה המערכת הבאה:

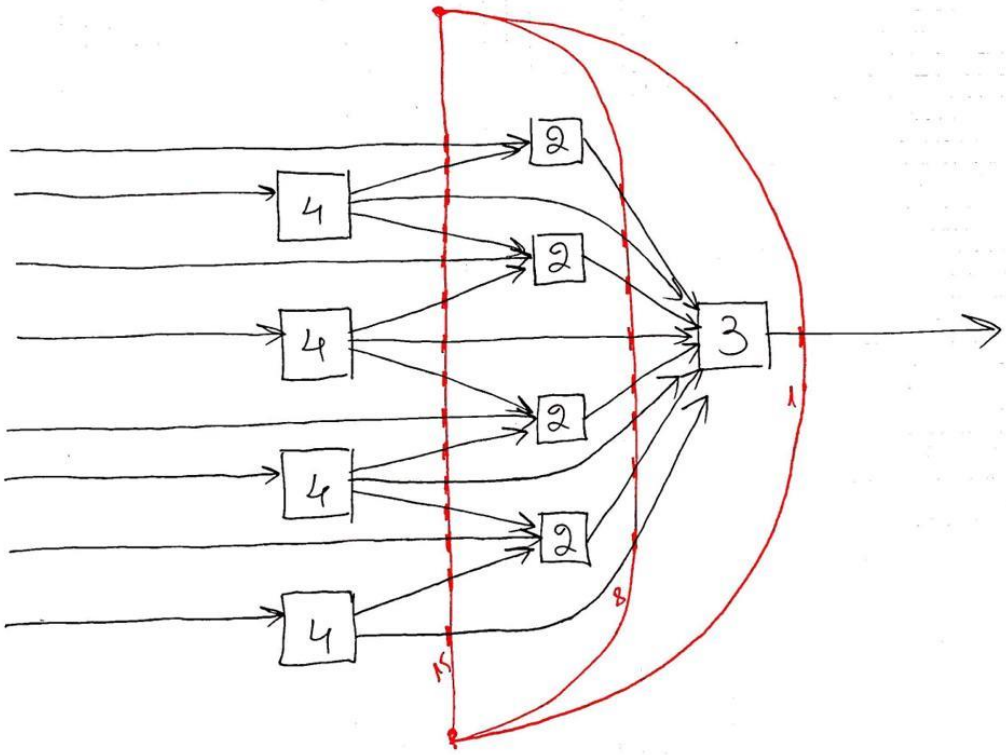


השהיית כל רכיב ב**ns** רשומה בתוך הקובייה.
מוצא המערכת הוא מהיחידה המרכזית (השהייה של **3ns**).
$t_{setup} = t_{pCQ}(FF) = 1ns$
נרצה לצנר את המערכת בצורה שתבטיח throughput מקסימלי (בעדיפות ראשונה) ו-latency מינימלי.
ניתן להניח כי הכניסות עומדות במשטר הזמנים הדינאמי.
מהו מספר הרגיסטרים שיבטיח תנאים אלו ומהו ה-**latency**?

א. Num =24 Latency =18ns
ב. Num =23 Latency =12ns
ג. Num =16 Latency =18ns
ד. Num =24 Latency =12ns
ה. Num =23 Latency =8ns

פתרון

צינור המערכת:



24 רגיסטרים עם דרגת צינור K=3

זמן המחזור המינימלי:

$$T_{min} = tp_{max} + t_{setup} + t_{pd}(FF) = 6ns$$

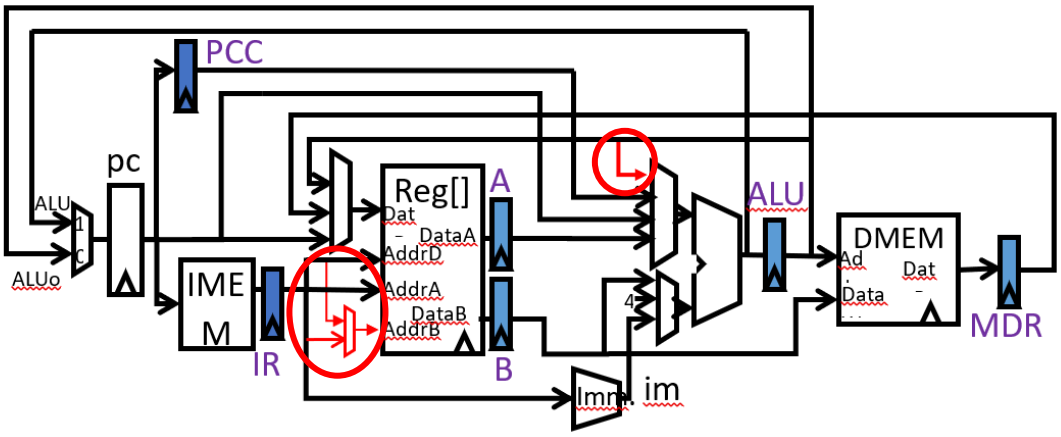
מפה נקבל:

$$\boldsymbol{latency = KT_{min} = 18ns}$$

נשים לב כי בחישוב זמן המחזור המינימלי הוספנו את $t_{pd}(FF)$ למרות שלא צינרנו גם את הכניסה. הסיבה לכך היא כי אנחנו מניחים שהרכיב שלנו יכול להיות משורשר אל רכיב אחר בעל רגיסטרים עם מאפיינים זהים ולכן עלינו להתחשב גם במצב של יחידה "4" התחומה בין שני רגיסטרים.

שאלה 13 (5 נקודות)

נתון מעבד ה-Multicycle Risc הבא אשר תומך בפקודות MAC (multiply and accumulate).
השינויים במעבד מוקפים במעגל.



מלבד קו הבקרה אשר שייך ל-Mux בכניסת ה- Register File לא נוספו קווי בקרה נוספים.
הפקודה מקודדת כפקודת R-type אך בעלת opcode ייחודי אשר שונה מכל הפקודות האחרות מסוג R-type.

פורמט פקודת ה-Mac הינו:

MAC rd, rs1, rs2

$$Reg[rd] = Reg[rd] + Reg[rs1] * Reg[rs2]$$

תזכורת:

Dispatch ROM 1		
Op	Name	Value
x33	R-type	6
x67	JAL	9
x63	BEQ	8
x03	LW	2
x23	SW	2

Dispatch ROM 2		
Op	Name	Value
x03	LW	3
x23	SW	5

אילו שינויים נצטרך לבצע במכונת המצבים של הבקר על מנת לתמוך בפקודה ה-MAC?
הניחו כי התמיכה בסיביות הבקרה של ה-muxes כבר נוספה לבקר.

ניתן ומומלץ להיעזר במכונת המצבים בדף הנוסחאות. **שימו לב כי מספרי המצבים הישנים (אלו אשר מופיעים בדף הנוסחאות ונלמדו בכיתה) אינם משתנים.**

א. הוספת שורה ב-Dispatch ROM 1, והוספת מצב חדש למכונת המצבים.
ב. הוספת Dispatch ROM חדש ושני מצבים חדשים.
ג. הוספת שורה ב-Dispatch ROM 1, הוספת Dispatch ROM חדש והוספת מצב חדש למכונת המצבים.
ד. לא נצטרך לבצע שינויים במכונת המצבים כי המעבד כבר תומך בכפל ובחיבור.
ה. הוספת שורה ב-Dispatch ROM 1, הוספת Dispatch ROM חדש ושני מצבים חדשים.

פתרון

תשובה ה.

נתון כי מכנות המצבים כבר תומכת בסיביות הבקרה של ה-Muxes אשר נוספו. בכדי לאפשר את התמיכה נצטרך להוסיף שורה בDR1 על מנת לסווג את הפקודה החדשה אל מצב 6. נצטרך להוסיף DR עבור מצב 6 שיפריד בין פקודות R רגילות לבין MAC. עבור פקודות MAC נעבור למצב נוסף בו נבצע את פעולה החיבור ולאחריו את שלב הWB. תשובה ג' אינה נכונה מכיוון שלא ניתן לעבור מהמצב החדש בחזרה אל מצב 7(מצב הWB של פקודות Rtype) ללא הוספה של DispatchROM נוסף, וזאת מכיוון שמספרי המצבים אינם בעלי הפרש של 1 לא ניתן להשתמש בקידום של המצב ב-1.

שאלה 14 (10 נקודות)

נתון מעבד Multi cycle RISC-V אשר תומך בחריגות אשר מפורטות בטבלה מטה. הטיפול בחריגות מבוצע על ידי שיטת קוד הגורם לחריגה. הפעולות אשר מבצעת מערכת ההפעלה לצורך הטיפול בכל סוג של חריגה מתוארות בטבלה. בנוסף, הטבלה מתארת מהו הקידוד עבור של כל חריגה.

סוג החריגה	קידוד	פעולה רצויה
חלוקה ב-0	1	השוואת רגיסטר המכנה ל-2
גלישה	4	השוואת הרגיסטרים אשר גורמים לגלישה ל-0

הניחו כי הגישה אל הרגיסטרים SCAUSE ו-SEPC מתבצעת בצורה זהה לשאר הרגיסטרים במערכת.
בשאלה זו הניחו כי ערכי הרגיסטרים בפעולות add מטופלים כמספרים בייצוג unsigned.
פעולות addi עובדות כרגיל.

השלימו את הקוד הבא כך שירוץ כהלכה ויתמוך בכל סוגי החריגות עבור קטע ה-main הנתון:

```
0x1AA0 0000          Main:   addi s0, x0, 4
0x1AA0 0004                  addi t0, x0, 4
0x1AA0 0008                  addi s1, x0, -1
0x1AA0 000C                  add s1, t0, s1
0x1AA0 0010                  div t0, s0, s1
0x1AA0 0014                  add s0, s0, t0
0x1AA0 0018          Exit:   j ra

0x1C09 0000  Interrupt handler:  addi sp, sp, -4
0x1C09 0004                     sw s0, _________
0x1C09 0008                     addi s0, x0, 1
0x1C09 000C                     addi s2, x0, 4
0x1C09 0010                     beq SCAUSE, s0, Div0
0x1C09 0014                     beq_________ , s2, ______
0x1C09 0018          Done:      lw s0, 0(sp)
0x1C09 001C                     addi sp, sp, 4
0x1C09 0020                     jr, _____________
0x1C09 0024          Div0:      addi _________   , x0, __________
0x1C09 0028                     j _____________
0x1C09 002C          Overflow:  add s1, x0, 0
0x1C09 0030                     addi _________ , __________ , __________
0x1C09 0034                     j Done
```

פתרון

```
0x1AA0 0000          Main:   addi s0, x0, 4
0x1AA0 0004                  addi t0, x0, 4
0x1AA0 0008                  addi s1, x0, -1
0x1AA0 000C                  add s1, t0, s1
0x1AA0 0010                  div t0, s0, s1
0x1AA0 0014                  add s0, s0, t0
0x1AA0 0018          Exit:   j ra

0x1C09 0000  Interrupt handler:  addi sp, sp, -4
0x1C09 0004                     sw s0, 0(sp)
0x1C09 0008                     addi s0, x0, 1
0x1C09 000C                     addi s2, x0, 4
0x1C09 0010                     beq SCAUSE, s0, Div0
0x1C09 0014                     beq SCAUSE, s2, Overflow
0x1C09 0018          Done:      lw s0, 0(sp)
0x1C09 001C                     addi sp, sp, 4
0x1C09 0020                     jr SEPC
0x1C09 0024          Div0:      addi s1, x0, 2
0x1C09 0028                     j Done
0x1C09 002C          Overflow:  addi s1, x0, 0
```

0x1C09 0030	addi t0, x0, 0
0x1C09 0034	j Done

שאלה 15 (10 נקודות)

נתון מעבד Pipeline RISC-V ללא hazard detection unit וללא forwarding בכלל.
למעבד אין יכולת לבצע flush כלל.

א. (2 נקודות) כיצד משפיעה העובדה שלמעבד אין יכולת לבצע flush על ביצוע פקודות branch?

ב. (8 נקודות) נתונה התוכנית:

```
1          addi t1, x0, 0
2          addi t2, t1, 2
3    loop: lw   t3, 0(s1)
4          lw   t4, 4(s1)
5          add t5, t3, t4
6          add t0, t0, t5
7          subi t2, t2, 1
8          bne  t1, t2, loop
9          add  t6, t6, s1
10         add t7, t7, s1
11         add  t8, t8, s1
```

מלאו את הטבלה הבאה (ראו דוגמא כיצד למלא בשורה הראשונה) כך שתתאר את מספר ה-nops אשר צריך להוסיף לקוד על מנת שירוץ באופן תקין. זכרו כי אין אפשרות לדעת מראש האם הקפיצה מתרחשת או לא.

בין פקודה מספר	לבין פקודה מספר	כמות nops
1	2	3

פתרון

א. אין אפשרות להניח שהקפיצה מתרחשת, צריך לחכות עד ההחלטה.

ב. 11.

```
1          addi t1, x0, 0
           nop
           nop
           nop
2          addi t2, t1, 2
3     loop: lw   t3, 0(s1)
4          lw   t4, 4(s1)
           nop
           nop
           nop
5          add t5, t3, t4
           nop
           nop
           nop
6          add t0, t0, t5
7          subi t2, t2, 1
           nop
           nop
           nop
8          bne  t1, t2, loop
           nop
           nop
           //nop- accepted if branch resolution in Mem
9          add  t6, t6, s1
10         add t7, t7, s1
11         add  t8, t8, s1
```

שאלה 16 (15 נקודות)

נגדיר 2 רצפים: רצף A – 1001, רצף B – 1100 (הרצפים מוגדרים משמאל לימין)

יש לממש מכונת מצבים סינכרונית מסוג מילי בעלת כניסה אחת ויציאה אחת, המפיקה 1 במוצא למשך מחזור שעון יחיד אם מתקבל בכניסה רצף A או רצף B. המימוש נדרש להיות מימוש מינימלי, כאשר עליכם לציין מהו מוצא המכונה בכל מעבר מצבים. הניחו כי במצב המכונה ההתחלתי המכונה קיבלה רצף של 0-ים.

דוגמא:

clk	1	2	3	4	5	6	7	8	9	10	11	12
in	0	1	0	0	1	1	0	0	1	1	0	0
out	0	0	0	0	1	0	0	1	1	0	0	1

פתרון