

## תוכן העניינים:

2	מבנה המחשב ותכן לוגי .....
2	מעבד העובד בשיטת הצנרת .....
2	מבוא לצנרת : .....
2	סיכום כללי : .....
5	מבנה הצנרת במעבד החד-מחזורי : .....
5	סיכום כללי : .....
9	שאלות : .....
13	תשובות סופיות : .....
14	סיכונים בצנרת : .....
14	סיכום כללי : .....
15	סיכונים נתונים בצנרת : .....
15	סיכום כללי : .....
21	שאלות : .....
27	תשובות סופיות : .....
28	סיכונים בקרה בצנרת : .....
28	סיכום כללי : .....
33	שאלות : .....
33	תשובות סופיות : .....
34	הערכת ביצועי מעבד : .....
34	סיכום כללי : .....
34	שאלות : .....
35	תשובות סופיות : .....
36	שאלות מסכמות עם מעבד הצנרת : .....
36	שאלות : .....
40	תשובות סופיות : .....

# מבנה המחשב ותכן לוגי

## מעבד העובד בשיטת הצנרת

### מבוא לצנרת:

#### סיכום כללי:

#### הגדרה:

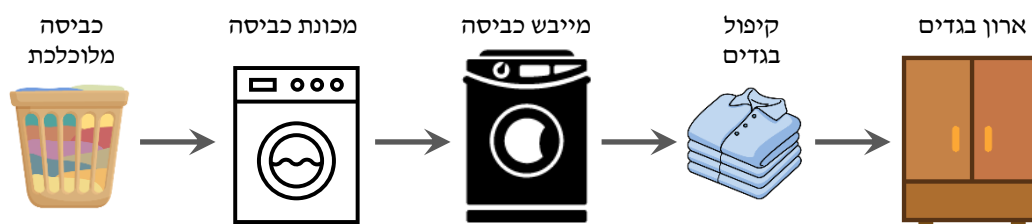
ה-pipelining (או בשמו בעברית : צְנֵרֶת/הצְנָרָה) הוא תהליך שבו שלבים יבוצעו במקביל זה לזה. הצנרה של שלבים קיימת כיום כמעט באופן אוניברסלי בכל מעבד וכן בתהליכים יומיומיים שונים.

#### אנלוגיה שתלווה אותנו לאורך הלמידה על מהות ההצנרה ותכונותיה:

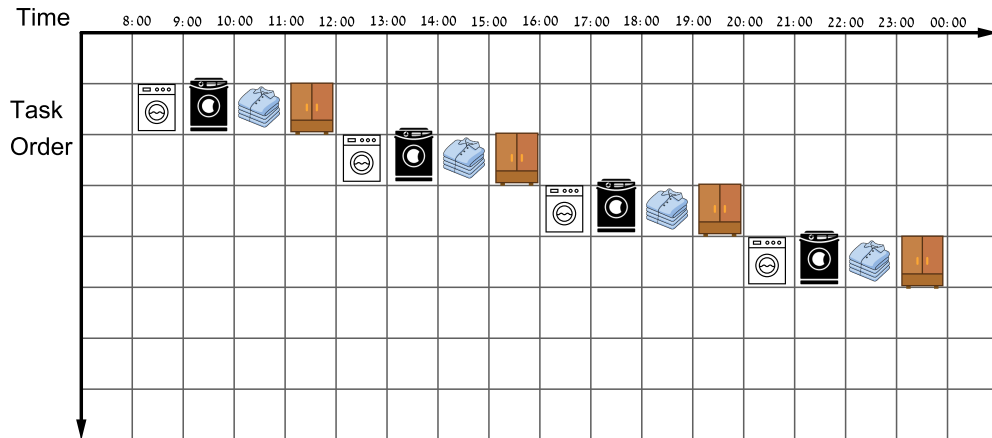
נתייחס משימת כביסה הכוללת 4 שלבים :

בהינתן כביסה מלוכלכת (קרי : משימת כביסה).

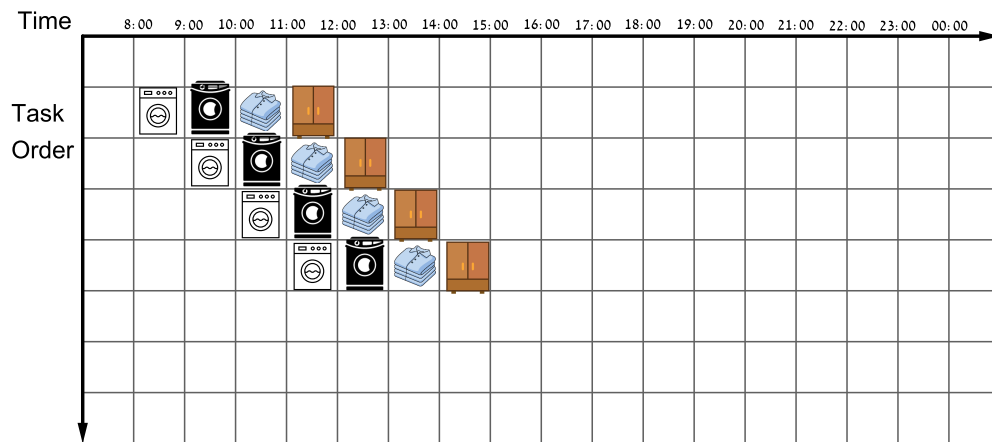
- שלב 1 - מכונת כביסה.
- שלב 2 – מייבש כביסה.
- שלב 3 – קיפול הבגדים.
- שלב 4 – השמת הכביסה בארון הבגדים.



### משך 4 משימות כביסה ללא הצנרה :



### משך 4 משימות כביסה כולל הצנרה :



### שלבים (stages) בצנרת:

בתהליך הצנרת, כל נקודת זמן מצביעה על השלב של משימה מסוימת והמיקום שלה בתהליך. תהליך הצנרת מחולק לשלבים וכאשר יש לנו משאבים שונים לכל שלב ניתן לבצע את ההצנרה.

### תובנה:

ה-pipelining משפר את ה-Throughput אך לא את ה-Latency מכיוון שהזמן לביצוע של כל משימה נשאר זהה, אך הזמן הכולל לביצוע של המשימות קטן.

## שיפור זמן הביצוע של תכנית שלמה ע"י הצנרה

בהנחה שמשכי הזמן של כל השלבים בתהליך של ביצוע משימה אחת זהים, ויש כמות גדולה של משימות זהות לבצע, נוכל לטעון כי תהליך ההצנרה משפר בקירוב את זמן הביצוע של תכנית פי מספר השלבים.

## יישום עיקרון ההצנרה במעבד ה-MIPS החד-מחזורי:

עבור מעבד עם מספר רב של שלבים ניתן לכתוב:

$$\text{Time between instruction (pipelined)} = \frac{\text{Time between instruction (non-pipelined)}}{\text{Number of stages}}$$

## איזון שלבי הצנרת:

היות ומשך זמן הביצוע של כל שלב בשלבי הצנרת אינו זהה, היחס המחושב הוא הגבול העליון מכיוון שישנם פרקי זמן בהם חומרה מהירה לא תבצע פעולה עד שהחומרה האיטית שלפניה תשלח לה מידע. לתופעה הזו קוראים **איזון שלבי הצנרת**.

**ככל שהסטייה בין משכי זמן הביצוע של כל שלב תהיה גדולה יותר, כך השיפור של הצנרת יהיה קטן יותר.**

## תיכנון ה-ISA בצורה ייעודית למימוש בהצנרה:

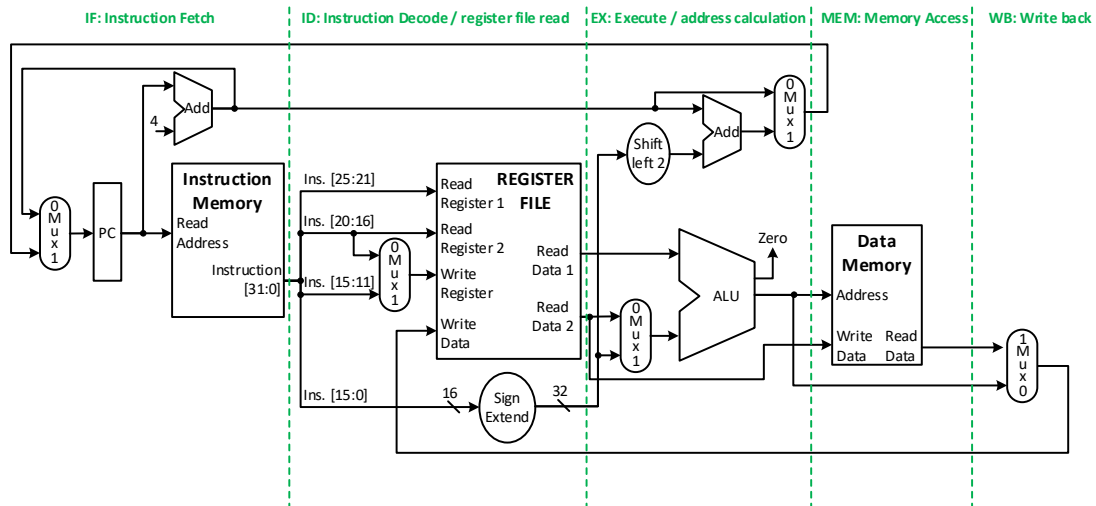
לאחר שרכשנו הבנה בסיסית לגבי תהליך ההצנרה, נוכל למנות 5 שיקולים בתיכנון ה-ISA של מעבדי ה-MIPS אשר מכוונים למימוש פקודות בצורה מקבילית:

- (1) כל פקודות ה-ISA הן באותו האורך.
- (2) למעבדי ה-MIPS יש מספר קטן של פורמטים (למדנו על 3 סוגים: R, I, J).
- (3) פקודות גישה לזיכרון בנויות כך שתחילה אנו מחשבים את כתובת הזיכרון בשלב ה-Execute ורק לאחר מכן ניגשים אליו.
- (4) מכיוון שהאופרנדים עצמם מסודרים בזיכרון כך שכל אחד מתחיל בגבול של המילה, לא נצטרך לגשת לזיכרון יותר מפעם אחת באותה הפקודה.
- (5) במקרים שבהם קיימת התנגשות בין פקודות כך שאחת דורשת מידע מפקודה שלפניה אשר נמצאת עוד בשלבי הצנרת, יהיה צורך לקדם (או לעקוף) חלק מהשלבים של הפקודה הראשונה על מנת לאפשר לפקודה המאוחרת להתבצע. לשיטה זו קוראים forwarding או bypassing ונלמד עליה בהמשך.

## מבנה הצנרת במעבד החד-מחזורי:

סיכום כללי:

מבנה הצנרת במעבד החד-מחזורי:

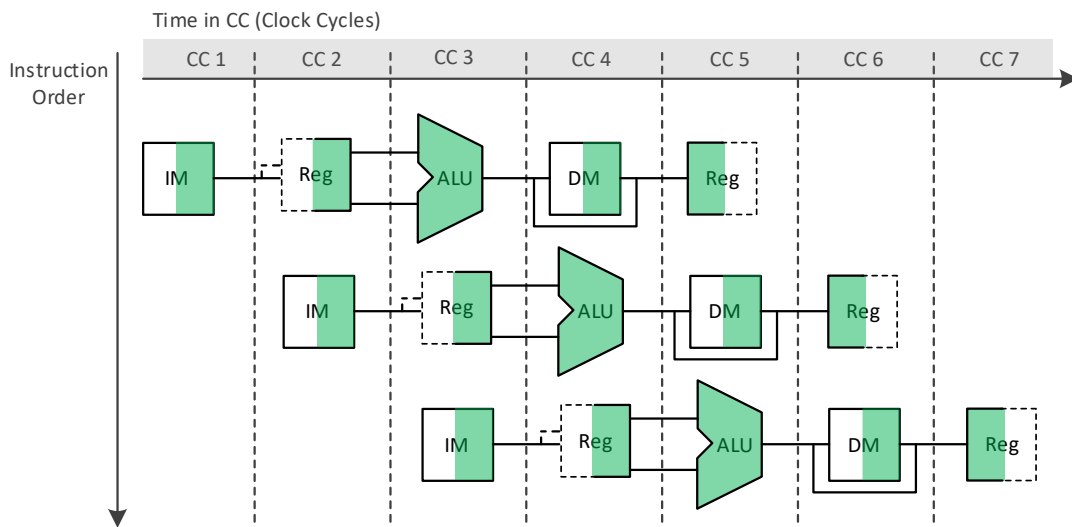


עיקרון תהליך הצנרת:

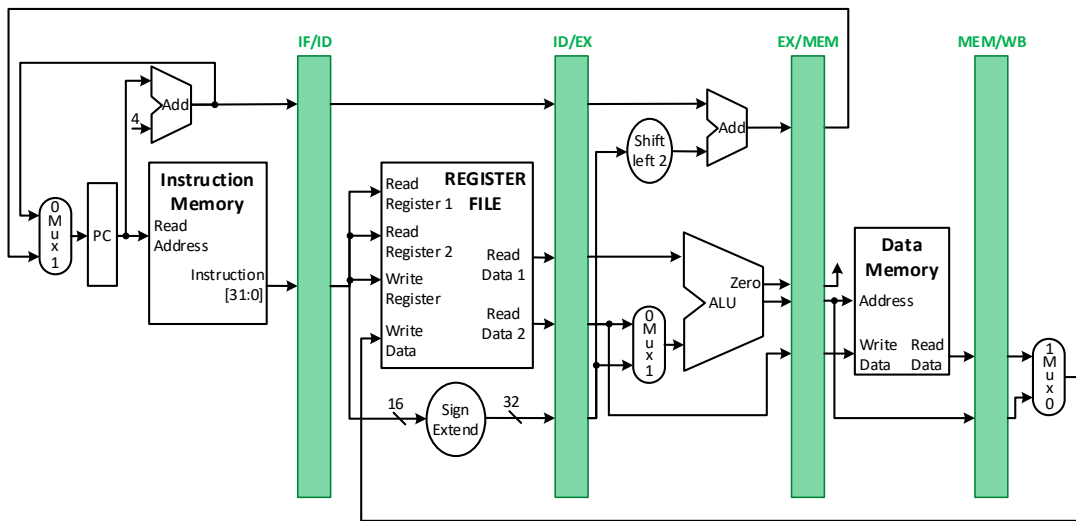
מידע זורם משמאל לימין ולא חוזר אחורנית.  
יחד עם זאת יש 2 מקרים יוצאים-מן-הכלל:

- (1) שלב ה-WB אשר מתקיים בפקודות מסוג R-Type ובפקודות lw ומעדכן את מקבץ האוגרים הנמצא באמצע הצנרת.
- (2) הגדלת ה-PC שבה הערך מעודכן בפקודות branch אחורנית משלב ה-MEM.

**ייצוג צנרת בתרשים פעימות שעון רבות (multiple-clock-cycle pipeline diagram):**



**ייצוג צנרת בתרשים פעימת שעון בודדת (single-clock-cycle pipeline diagram):**



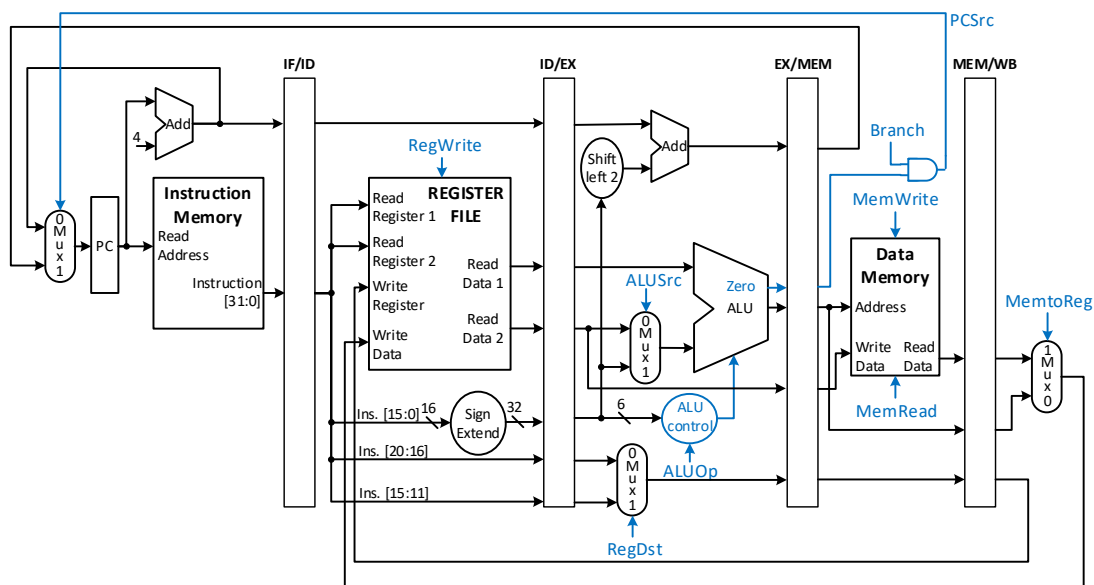
האוגרים של הצנרת נקראים לפי שני השלבים שהם מפרידים ביניהם:

- IF/ID - מפריד בין שלב ה-Instruction Fetch ושלב ה-Instruction Decode וגודלו 64 סיביות.
- ID/EX - מפריד בין שלב ה-Instruction Decode ושלב ה-Execute וגודלו 128 סיביות.
- EX/MEM - מפריד בין שלב ה-Execute ושלב ה-Memory וגודלו 97 סיביות.
- MEM/WB - מפריד בין שלב ה-Memory ושלב ה-Write Back וגודלו 64 סיביות.
- אפשר להסתכל על אוגר ה-PC בתור 'האוגר הראשון' מכיוון שהוא מזין את השלב הראשון ה-IF.

**המקרים בהם יש להעביר נתונים ממחזור שרון מסוים למחזור שרון מאוחר יותר שאינו עוקב:**

- בפקודת lw יש להעביר את כתובתו של האוגר rt משלב ה-ID עד לשלב ה-WB. הערך של rt יעבור ממחזור השרון השני מאוגר הצנרת IF/ID, דרך שאר האוגרים עד לאוגר הצנרת MEM/WB.
- בפקודת sw יש להעביר את ערכו של rt (\$rt) משלב ה-ID לשלב ה-MEM. גם כאן הערך של rt יעבור ממחזור השרון השני מאוגר הצנרת IF/ID דרך ID/EX ויסיים באוגר הצנרת EX/MEM.
- בפקודת beq הערך של PC + 4 נכתב במחזור השרון הראשון אך יש לעשות בו שימוש רק במחזור השרון השלישי. לכן ערך זה יעבור מאוגר הצנרת IF/ID לאוגר ID/EX.

**קווי הבקרה של הצנרת:**



## חלוקת קווי בקרה:

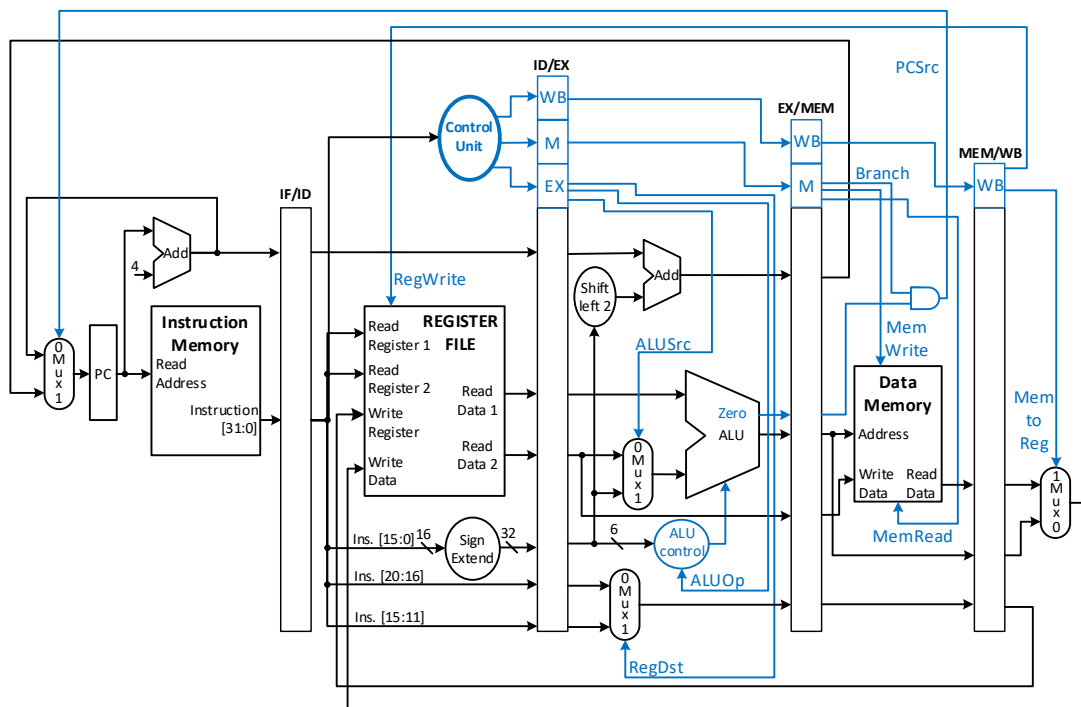
היות וכל אוגר צנרת נכתב בכל מחזור שעון, נוכל לחלק את הכתיבה של קווי הבקרה לחמשת שלבי הצנרת באופן הבא:

- (1) שלב ה-IF: יכול לזכות את קווי הבקרה לקריאת הפקודה מהזיכרון (כמו במעבד חד-מחזורי).
- (2) שלב ה-ID: יכול לזכות רק את הכתיבה למקבץ האוגרים (כמו במעבד חד-מחזורי).
- (3) שלב ה-EX: יכול לזכות כתיבה ל-RegDst, ל-ALUOp ול-ALUSrc.
- (4) שלב ה-MEM: יכול לזכות כתיבה ל-Branch, ל-MemRead ול-MemWrite.
- (5) שלב ה-WB: יכול לזכות כתיבה ל-MemtoReg ול-RegWrite.

היות ומשמעות קווי הבקרה זהה לשל המעבד החד-מחזורי, נוכל לשמור על אותם הערכים כפי שלמדנו בעבר. נוכל לקבץ קווי הבקרה לקבוצות לפי דרגת הצנרת בה כותבים אליהם:

Ins.	Execution			Memory Access			Write Back	
	RegDst	ALUOp	ALUSrc	Branch	MemRead	MemWrite	RegWrite	MemtoReg
R-format	1	10	0	0	0	0	1	0
lw	0	00	1	0	1	0	1	1
sw	X	00	1	0	0	1	0	X
beq	X	01	0	1	0	0	0	X

## תרשים כולל – מעבד צנרת עם קווי הבקרה:





## שאלות:

1) לפניכם מספר משפטים. קבעו לגבי כל אחד האם הוא נכון או שגוי ונמקו את טענתכם.

א. במעבד העובד בשיטת הצנרת מידע תמיד יזרום משמאל לימין, כלומר לא ילך 'אחורנית' בתרשים המעבד החד-מחזורי.

ב. במודל המעבד שעובד בשיטת הצנרת, לא תהיינה בעיות וסיכונים כלשהם כל עוד המידע זורם משמאל לימין.

בתרשים פעימות שעון רבות, ראינו כי הדגשה של מחצית בלוק משמעה שימוש במחצית ממחזור השעון. הדגשה של חצי שמאלי מתארת שימוש במחצית מחזור השעון הראשונה והדגשה של המחצית הימנית מתארת שימוש במחצית מחזור השעון השנייה.

ג. בהתאם לכך, ישנן פקודות בהן בלוק ה-IM (Instruction Memory) יודגש בחלקו השמאלי ופקודות בהן בלוק זה יודגש בחלקו הימני.

ד. קיימות פקודות בהן לא נדגיש את בלוק ה-ALU.

ה. במודל מעבד הצנרת היסודי (הבנוי על מודל המעבד החד-מחזורי), לא קיימות פקודות בהן בלוק ה-Reg (מקבץ האוגרים) יודגש במלואו.

בתרשים פעימת שעון בודדת למדנו כי ישנם ארבעה אוגרי צנרת.

ו. הסיבה לקיום ארבעה ולא חמישה אוגרים היא שניתן להסתכל על אוגר ה-PC כאל אוגר צנרת חמישי בכך שהוא מאחסן את כתובת הפקודה הבאה שעוברת במחזור השעון הראשון של פקודה הנכנסת לצנרת והוא נכתב מחדש בכל מחזור שעון.

ז. ההבדל בין אוגר ה-PC לשאר אוגרי הצנרת הוא שאוגר זה מעביר רק קווי נתונים (כתובת הפקודה הבאה בזיכרון הפקודות) בעוד ששאר אוגרי הצנרת מעבירים את ערכי קווי הנתונים ואת ערכי קווי הבקרה לשלב הבא.

ח. לכל האוגרים אותה מידה והיא 128 ביטים (4 מילים) היות וכל אוגר צריך להכיל את מילת הפקודה ( $Ins[31:0]$ ), את המידע האגור בשני אוגרים ממקבץ האוגרים ( $\$rs, \$rt$ ) ועוד מילה עבור ערכי קווי הבקרה המחלחלים משלב לשלב.

מודל מעבד העובד בשיטת הצנרת כולל קווי בקרה הניתנים לחלוקה לפי השלב בצנרת אליו הם נכתבים. אלו מועברים משלב לשלב כדי לאפשר לצנרת לבצע את הפקודה המתקבלת.

ט. כל ערכי קווי הבקרה הנוגעים ליחידת הזיכרון (והם: MemWrite, MemRead, MemtoReg) ורק הם ייכתבו בשלב הרביעי של הצנרת, שלב ה-MEM.

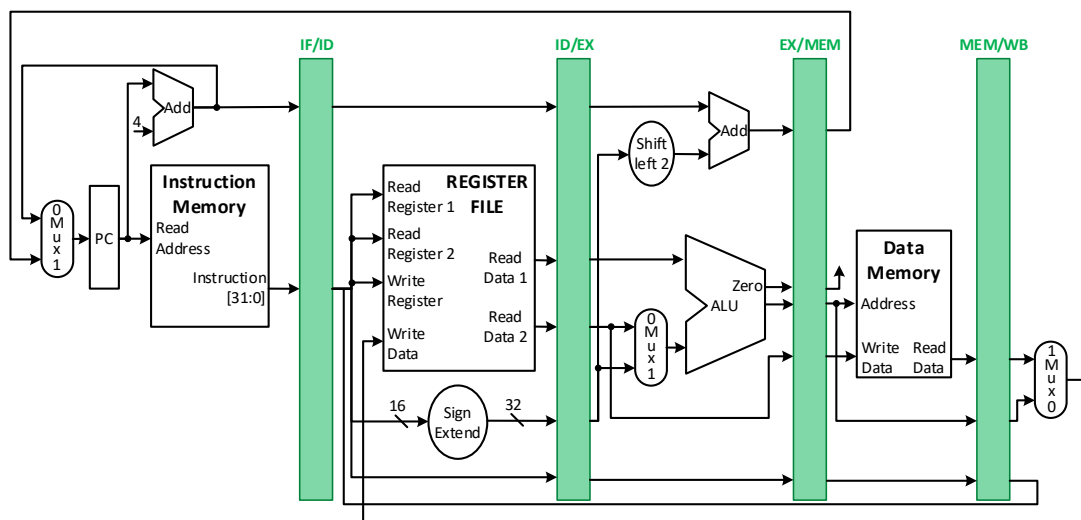
י. היות ושלבי ה-IF וה-ID (שני השלבים הראשונים בצנרת) זהים בכל סוגי הפקודות ב-ISA, אין צורך להעביר מהם את ערכי קווי הבקרה המתאימים לשלבים הבאים בצנרת.

יא. ערכי קווי הבקרה עוברים דרך אוגרי הצנרת מכיוון שהם מכתיבים את הפעילות שיש לבצע עם מידע הנמצא באוגרים אלו.

## (2) נתונה התכנית הבאה :

```
lw $8, 40($7)
lw $9, 40($7)
add $20, $16, $12
add $22, $17, $13
sub $31, $8, $6
sw $10, -100($7)
```

- כמה מחזורי שעון יצטרך מעבד העובד הצנרת כדי לבצע את התכנית?
- איזו פקודה תהיה בכל דרגה בצנרת במחזור השעון השלישי? החמישי? השביעי?



3) במחזור שעון מסוים, קווי הבקרה של מעבד העובד בשיטת הצנרת מראים את הערכים הבאים:

Execution			Memory Access			Write Back	
RegDst	ALUOp	ALUSrc	Branch	MemRead	MemWrite	RegWrite	MemtoReg
1	10	0	0	0	1	1	1

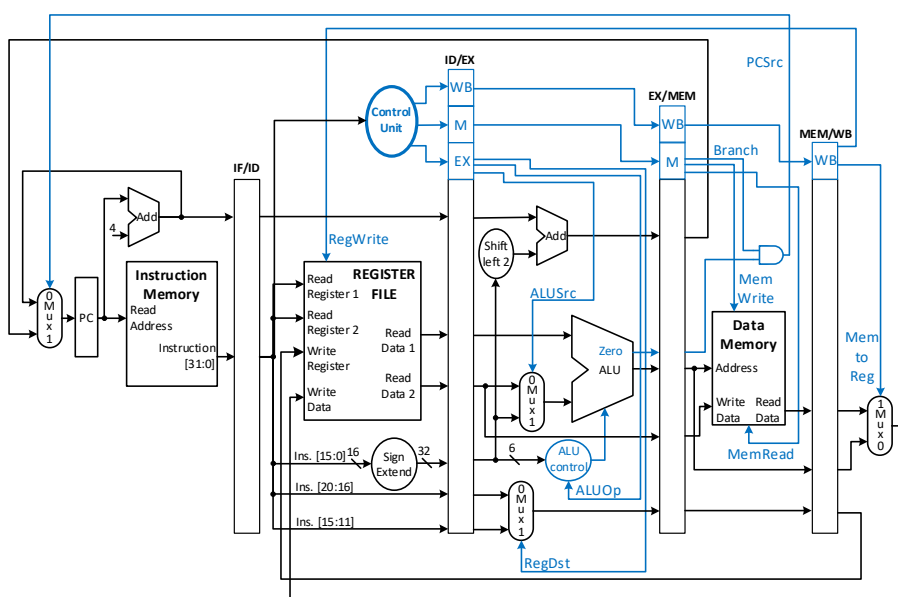
על סמך ערכים אלו, קבעו אלו מבין הפקודות הבאות עשויות להיות בצנרת בשלבים 3-5. נמקו.

- א. שלב EX: `lw $4, 400($5)` שלב MEM: `add $8, $9, $18` שלב WB: `or $7, $10, $0`
- ב. שלב EX: `add $4, $5, $6` שלב MEM: `add $8, $9, $10` שלב WB: `and $7, $10, $0`
- ג. שלב EX: `add $4, $5, $6` שלב MEM: `sw $8, 92($10)` שלב WB: `lw $7, 400($20)`
- ד. שלב EX: `sub $4, $5, $6` שלב MEM: `sw $8, 0($18)` שלב WB: `beq $7, $10, 0x49`

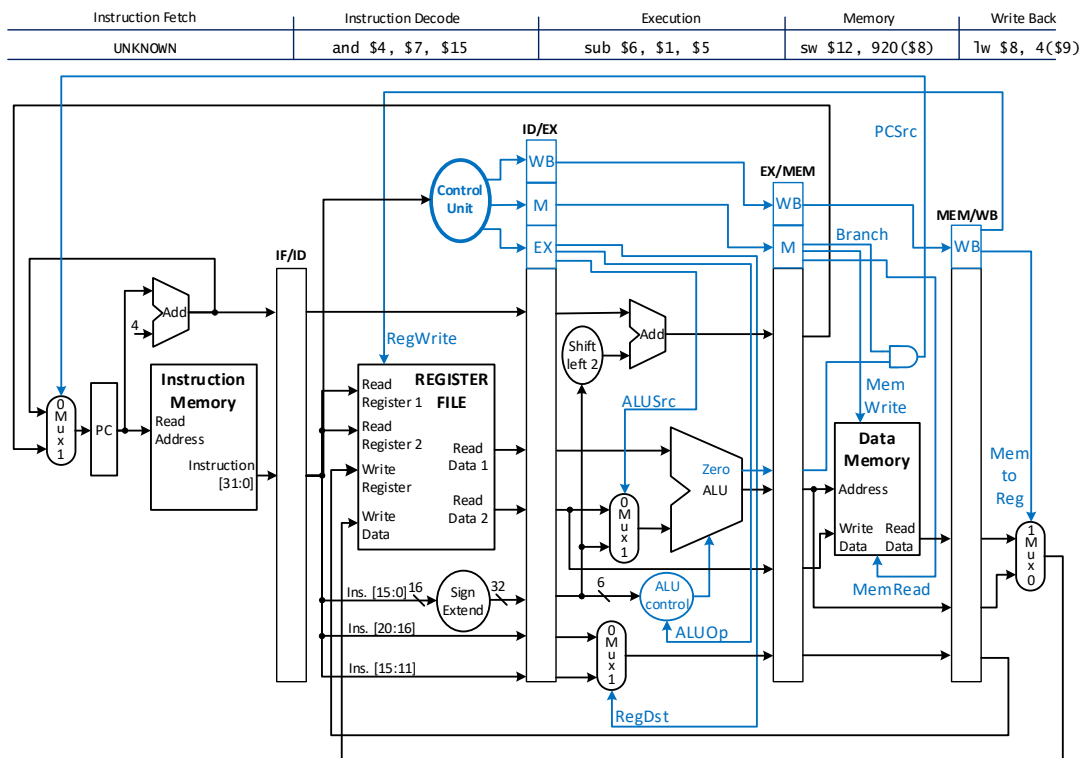
טבלת עזר:

Ins.	Execution			Memory Access			Write Back	
	RegDst	ALUOp	ALUSrc	Branch	MemRead	MemWrite	RegWrite	MemtoReg
R-format	1	10	0	0	0	0	1	0
lw	0	00	1	0	1	0	1	1
sw	X	00	1	0	0	1	0	X
beq	X	01	0	1	0	0	0	X

סרטוט עזר:



4) לפניכם תרשים מעבד העובד בשיטת הצנרת עבור מחזור שעון מסוים.



א. מה הם ערכי קווי הבקרה הנכתבים לכל אחד מן השלבים : EX, MEM, WB?

נתונים כעת משכי הזמן של שלבי הצנרת :

IF	ID	EX	MEM	WB
250ps	300ps	450ps	750ps	300ps

- מהו זמן מחזור השעון לפי המודל החד-מחזורי ולפי מודל הצנרת?
- כמה זמן ייקח לבצע את פקודת lw לפי כל מודל?
- נניח וניתן לפצל את אחד משלבי הצנרת ל-3 חלקים, כל חלק שווה באורכו. איזה שלב הייתם מפצלים וכיצד פיצול זה ישפיע על תוצאות סעיפים ב-ג?
- האם פיצול כמו המתואר בסעיף הקודם יגדיל את התפוקה (throughput) של מעבד הצנרת?

## תשובות סופיות:

- (1) טענות נכונות: ב', ד', ו', י', יא'. טענות שגויות: א', ג', ה', ז', ח', ט'.  
ראו הסברים מפורטים בסרטון הוידאו באתר גול.
- (2) א. 10 מחזורי שעון.  
ב. מחזור שעון שלישי:  

$$\underline{IF}: \text{add } \$20, \$16, \$12 \quad \underline{ID}: \text{lw } \$9, 40(\$7) \quad \underline{EX}: \text{lw } \$8, 40(\$7)$$

$$\underline{MEM}: \langle \rangle \quad \underline{WB}: \langle \rangle$$

מחזור שעון חמישי:

$$\underline{IF}: \text{sub } \$31, \$8, \$6 \quad \underline{ID}: \text{add } \$22, \$17, \$13 \quad \underline{EX}: \text{add } \$20, \$16, \$12$$

$$\underline{MEM}: \text{lw } \$9, 40(\$7) \quad \underline{WB}: \text{lw } \$8, 40(\$7)$$

מחזור שעון שביעי:

$$\underline{IF}: \langle \rangle \quad \underline{ID}: \text{sw } \$10, -100(\$7) \quad \underline{EX}: \text{sub } \$31, \$8, \$6$$

$$\underline{MEM}: \text{add } \$22, \$17, \$13(\$7) \quad \underline{WB}: \text{add } \$20, \$16, \$12$$
- (3) תשובה ג.
- (4) א.  $EX = 1100, MEM = 010, WB = 11$   
 ב. חד מחזורי:  $CCT = 2050 \text{ ps}$ , צנרת:  $CCT = 750 \text{ ps}$ .  
 ג. חד מחזורי:  $T(lw) = 2050 \text{ ps}$ , צנרת:  $T(lw) = 3750 \text{ ps}$ .  
 ד. נפצל את שלב ה-MEM ל-3 חלקים שאורך כל אחד הוא 250ps.  
 חד מחזורי:  $T(lw) = 2050 \text{ ps}$ , צנרת:  $T(lw) = 3150 \text{ ps}$ .  
 ה. התפוקה תגדל מכיוון שכעת כל פעולה מתבצעת במשך 450ps במקום 750ps.

## סיכונים בצנרת:

### סיכום כללי:

#### סיכונים בתהליך הצנרת:

ישנם 3 סיכונים כאשר מיישמים תהליך של מיקבול שלבים בביצוע פקודות:

- סיכוני מבנה (Structural Hazards) או פשוט Hazards :  
סיכונים הנובעים מכך ששתי פקודות רצופות דורשות את אותה החומרה במחזור שעון מסוים.
- סיכוני נתונים (Data Hazards) :  
סיכונים הגוררים עיכוב של הצנרת מכיוון שהמידע הנחוץ מפקודה קודמת עוד לא חושב/הושלם.
- סיכוני בבקרה (Control Hazards) :  
סיכונים הנובעים מכך שקווי הבקרה נקבעים עקב תוצאות חישוב (execute) שטרם בוצע, כגון בפקודת beq.

#### סיכוני מבנה (Structural Hazard) :

במקרה שבו קיימות שתי פקודות (לרוב סמוכות) אשר במחזור שעון מסוים דורשות את אותה החומרה, לא נוכל לבצע את המיקבול של תהליך ההצנרה. החומרה לא מסוגלת לתמוך בצירוף פקודות שכזה עקב הדרישה לשימוש באותו המשאב במחזור שעון נתון. כפי שציינו בעבר, מעבדי ה-MIPS תוכננו כך שיהיו ניתנים להצנרה, ולכן ניתן בקלות להימנע מסיכון מסוג זה. עקב כך לא נעסוק בסיכון זה במהלך הלמידה שלנו על מעבדי ה-MIPS.

#### סיכוני נתונים (Data Hazard) :

במקרה שבו יש לעצור את ההצנרה מכיוון ששלב מסוים תלוי בתוצאת חישוב של שלב קודם אשר טרם בוצע נאמר כי מדובר בסיכון הנובע מחילחול המידע עצמו בשלבי הצנרת.

#### סיכון בקרה (Control Hazard) :

סוג זה של סיכון נובע מכך שיש לקבל החלטה על סמך תוצאות פקודה שנמצאת בצנרת. הדוגמה הנפוצה ביותר היא פקודת beq, שבה רק לאחר קבלת תוצאת הסיבית zero ניתן לדעת מה תהיה הפקודה הבאה.

## סיכוי נתונים בצנרת:

### סיכום כללי:

#### קידום/העברת ערך (Forwarding):

כאשר מידע מוכן בסיום שלב 3 (ה-EX) ויש בו צורך בפקודות סמוכות הנמצאות בצנרת אך הוא טרם נכתב חזרה למקבץ האוגרים בשלב ה-WB, נוכל לקדם אותו ע"י העברה שלו לפקודות הבאות בטרם הפקודה הנוכחית כותבת אותו למקבץ האוגרים. לפעולה הזו קוראים **קידום או העברה** (באנגלית: Forwarding).

#### מימוש forwarding במעבד הצנרת:

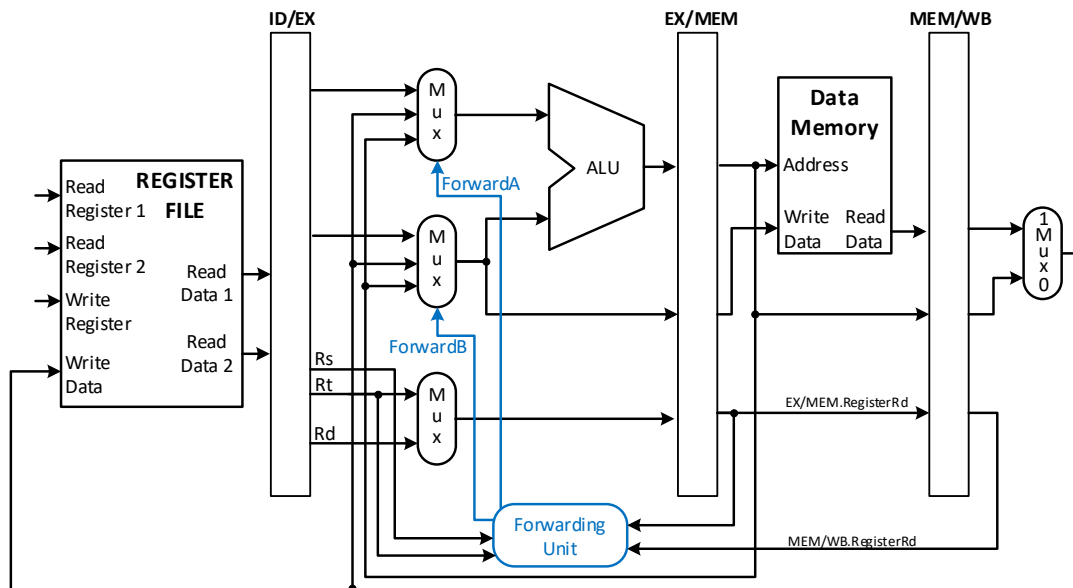
נאמץ את הסימון הבא: "ID/EX.RegisterRs" כדי לציין את מספר האוגר שהערך שלו נמצא באוגר הצנרת ID/EX והוא מיועד להיכנס כהאופרנד הראשון ליחידת ה-ALU. באותו אופן:

שם האוגר	משמעות
ID/EX.RegisterRs	האוגר בתוך אוגר הצנרת ID/EX שמיועד להיכנס לאופרנד הראשון ב-ALU.
ID/EX.RegisterRt	האוגר בתוך אוגר הצנרת ID/EX שמיועד להיכנס לאופרנד השני ב-ALU.
EX/MEM.RegisterRs	האוגר בתוך אוגר הצנרת EX/MEM שמיועד להיכנס לאופרנד הראשון ב-ALU.
EX/MEM.RegisterRt	האוגר בתוך אוגר הצנרת EX/MEM שמיועד להיכנס לאופרנד השני ב-ALU.
EX/MEM.RegisterRd	האוגר בתוך אוגר הצנרת EX/MEM שאליו תיכתב התוצאה של פקודת ה-ALU.
MEM/WB.RegisterRd	האוגר בתוך אוגר הצנרת MEM/WB שאליו תיכתב התוצאה של פקודת ה-ALU.

#### תנאים לקיום סיכוי נתונים:

- תנאי 1a:  $EX/MEM.RegisterRd = ID/EX.RegisterRs$
- תנאי 1b:  $EX/MEM.RegisterRd = ID/EX.RegisterRt$
- תנאי 2a:  $MEM/WB.RegisterRd = ID/EX.RegisterRs$
- תנאי 2b:  $MEM/WB.RegisterRd = ID/EX.RegisterRt$

## מבנה יסודי של יחידת ההעברה:



## סיכום ערכי קווי הבקרה לרכיבי ה-MUX בכניסות ל-ALU:

הסבר	אוגר צנרת	ערך בקרה
האופרנד הראשון ל-ALU מגיע מאוגר הצנרת ID/EX	ID/EX	ForwardA = 00
האופרנד הראשון ל-ALU מגיע מאוגר הצנרת EX/MEM	EX/MEM	ForwardA = 10
האופרנד הראשון ל-ALU מגיע מאוגר הצנרת MEM/WB	MEM/WB	ForwardA = 01
האופרנד השני ל-ALU מגיע מאוגר הצנרת ID/EX	ID/EX	ForwardB = 00
האופרנד השני ל-ALU מגיע מאוגר הצנרת EX/MEM	EX/MEM	ForwardB = 10
האופרנד השני ל-ALU מגיע מאוגר הצנרת MEM/WB	MEM/WB	ForwardB = 01



**סיכום ערכי קווי הבקרה לקיום סיכוני נתונים עבור יחידת ה-Forward :**

• סיכון נתונים בשלב ה-EX :

```
if      ( (EX/MEM.RegWrite)
and (EX/MEM.RegisterRd ≠ 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRs) )
then    ForwardA = 10
```

```
if      ( (EX/MEM.RegWrite)
and (EX/MEM.RegisterRd ≠ 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRt) )
then    ForwardB = 10
```

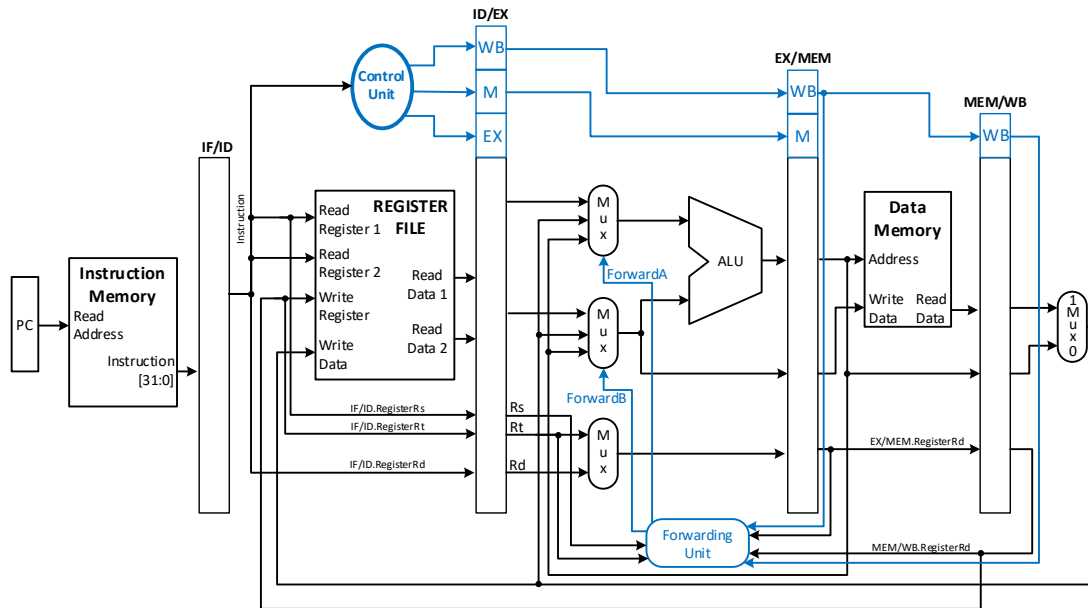
• סיכון נתונים בשלב ה-MEM :

```
if      ( (MEM/WB.RegWrite)
and (MEM/WB.RegisterRd ≠ 0)
and not ( (EX/MEM.RegisterWrite)
and (EX/MEM.RegisterRd ≠ 0)
and (EX/MEM.RegisterRd ≠ ID/EX.RegisterRs) )
and (MEM/WB.RegisterRd = ID/EX.RegisterRs) )
then    ForwardA = 01
```

```
if      ( (MEM/WB.RegWrite)
and (MEM/WB.RegisterRd ≠ 0)
and not ( (EX/MEM.RegisterWrite)
and (EX/MEM.RegisterRd ≠ 0)
and (EX/MEM.RegisterRd ≠ ID/EX.RegisterRt) )
and (MEM/WB.RegisterRd = ID/EX.RegisterRt) )
then    ForwardB = 01
```

- אין סיכון נתונים בשלב ה-WB כי הכתיבה והקריאה למקבץ האוגרים מתבצעת באותו מחזור השעון. (קיים Forwarding פנימי בתוך מקבץ האוגרים אבל הוא לא מעניינינו).

## מבנה מסכם של יחידת ההעברה:



## מקרה ה-Load Use:

כאשר פקודת `lw` מופיעה לפני פקודה שמבקשת את המידע המובא מהזיכרון, לא ניתן לבצע את שתייהן ללא כל התערבות. זאת כיוון שהנתון המובא מהזיכרון לאוגרי הצגרת בפקודת ה-`lw` מתבצע רק בסוף מחזור השעון הרביעי לפקודה בעוד שהקריאה לנתון בפקודה הבאה מתבצעת המחזור השני. לכן קריאה לנתון ע"י הפקודה המאוחרת לא אפשרית כי הוא טרם הגיע לאוגר עוד מפקודת ה-`lw`. למקרה זה אנו קוראים **Load Use**.

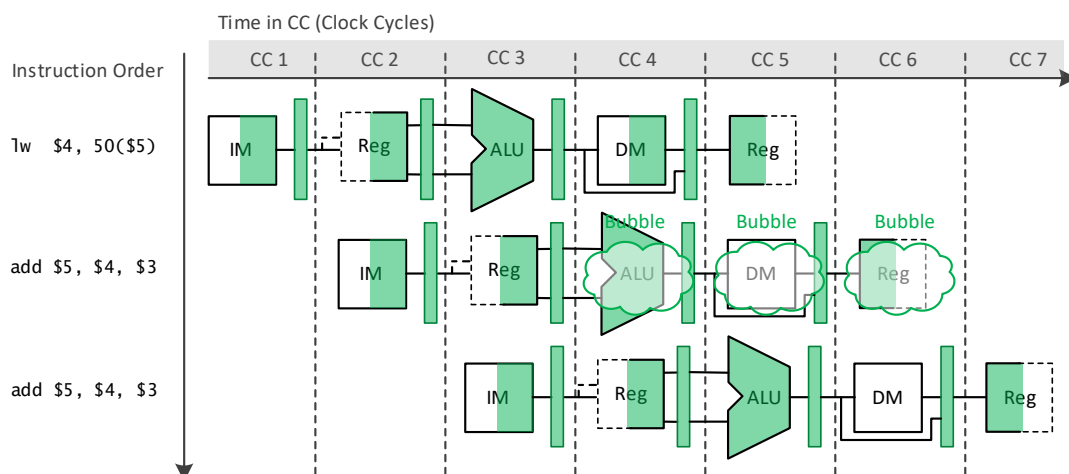
## יחידת איתור סיכונים - (HDU) Hazard Detection Unit:

נמקם HDU בשלב ה-ID בו נחליט אם להכניס עיכוב (stall) לצגרת לפני שהפקודה הבאה תיכנס. על היחידה לפעול כאשר מתקבלת פקודת `lw`, ולבדוק האם בפקודה שאחריה התוכן של אוגר היעד (אליו תיכתב המילה מהזיכרון) יידרש באחת מהכניסות ל-ALU.

```

if      ( (ID/EX.MemRead)
and    ( (ID/EX.RegisterRt = IF/ID.RegisterRs)
or      (ID/EX.RegisterRt = IF/ID.RegisterRt) ) )
then    stall the pipeline
    
```

כדי שהפקודה בשלב ה-ID תתעכב במחזור שעון אחד עלינו לעכב גם את הפקודה שנמצאת בשלב ה-IF. כדי לבצע זאת עלינו לדאוג שה-PC לא יקודם ובכך אוגר הצנרת ID/IF לא יעדכן את הערכים שלו במשך מחזור השעון הזה. במקרה זה כל האוגרים שבשלב ה-ID במקבץ האוגרים עדיין ייקראו ללא שום שינוי. כדי לאפשר לשאר הצנרת לבצע משהו נעביר את כל קווי הבקרה שלהם ל-0, פעולה זו נקראת הכנסת nop לצנרת (do nothing או nop = no operation).



### חציית מקבץ האוגרים:

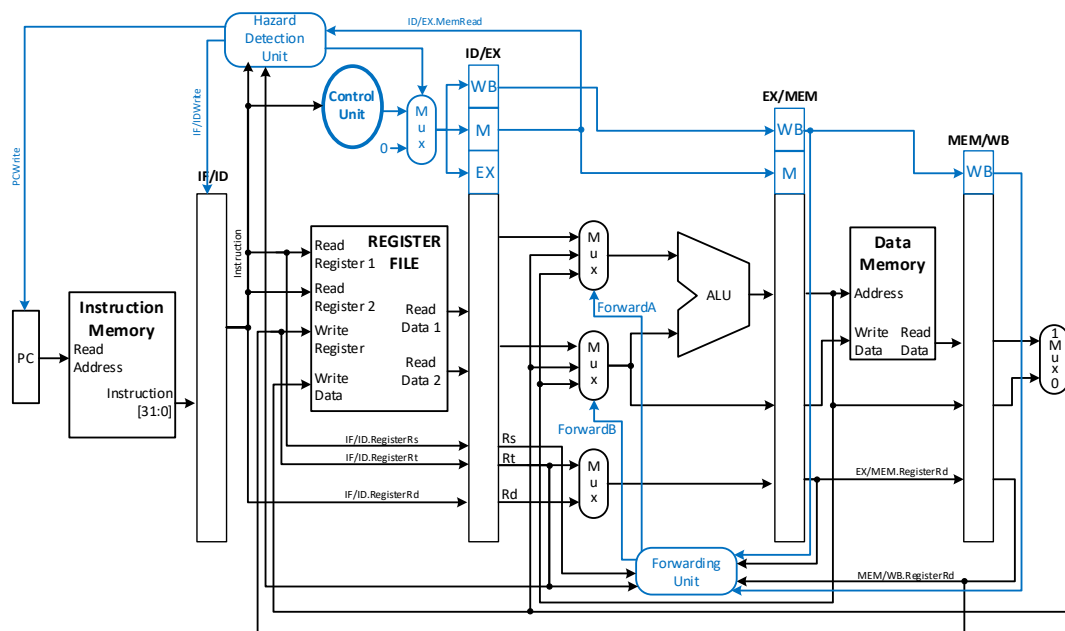
במידה ובשאלה מצוין כי ניתן לכתוב ולקרוא ממקבץ באוגרים באותו מחזור שעון (כגון שמקבץ האוגרים בנוי חומרתית בדרך המאפשרת כתיבה אליו במחצית מחזור שעון הראשון קריאה ממנו במחצית מחזור השעון שני) אז נאמר כי במקבץ האוגרים קיימת חומרה האחראית על חציית מקבץ האוגרים.

### מה יחידת איתור הסיכונים תבצע כאשר היא מזהה סיכון נתונים?

כדי לממש את עיכוב הצנרת על היחידה לבצע 3 דברים:

- לאפס את כל ערכי קווי הבקרה שעוברים לשלב הבא בצנרת. נבצע ע"י הוספת mux שבורר בין העברת ערכי קווי הבקרה מיחידת הבקרה לבין הקבוע 0.
- למנוע עידכון אוגר הצנרת IF/ID ע"י כיבוי ביט הבקרה IF/IDWrite. בכך, הפקודה שלאחר lw תבצע שוב את ה-ID במחזור שעון נוסף.
- למנוע עידכון של אוגר ה-PC ע"י כיבוי ביט הבקרה PCWrite. בכך, המידע המובא מהזיכרון בשלב ה-IF יתבצע שנית.

מבנה מעבד צנרת עם יחידת ההעברה ויחידת ה-HDU:



## שאלות:

### שאלות כלליות:

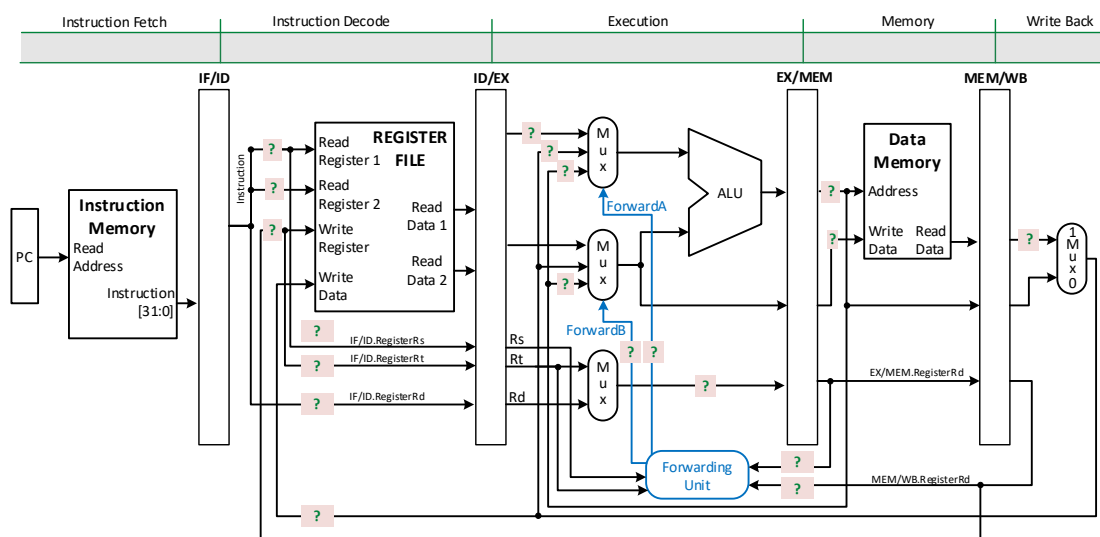
- (1) לפניכם קטע קוד המיוחס למעבד MIPS העובד בשיטת הצנרת. הניחו כי במקבץ האוגרים קיימת חצייה פנימית.

```
add $20, $18, $18
sub $18, $16, $20
add $15, $20, $21
sub $7, $19, $20
```

- א. זהו וסמנו את כל סיכוני הנתונים.  
 ב. אלו סיכונים ניתן למנוע בעזרת יחידת ההעברה?  
 ציינו את ביצוע קידום המידע הנדרש בכל מקרה.  
 ג. (1) במידה וקיימת יחידת העברה במעבד, האם יהיה צורך בהכנסת פקודות nop? אם כן כמה והיכן?  
 (2) במידה ולא קיימת יחידת העברה במעבד, האם יהיה צורך בהכנסת פקודות nop? אם כן כמה והיכן?

- (2) בתרשים שלפניכם מופיעה סכמת פעימת שעון ריקה ונתון הקטע הקוד הבא:

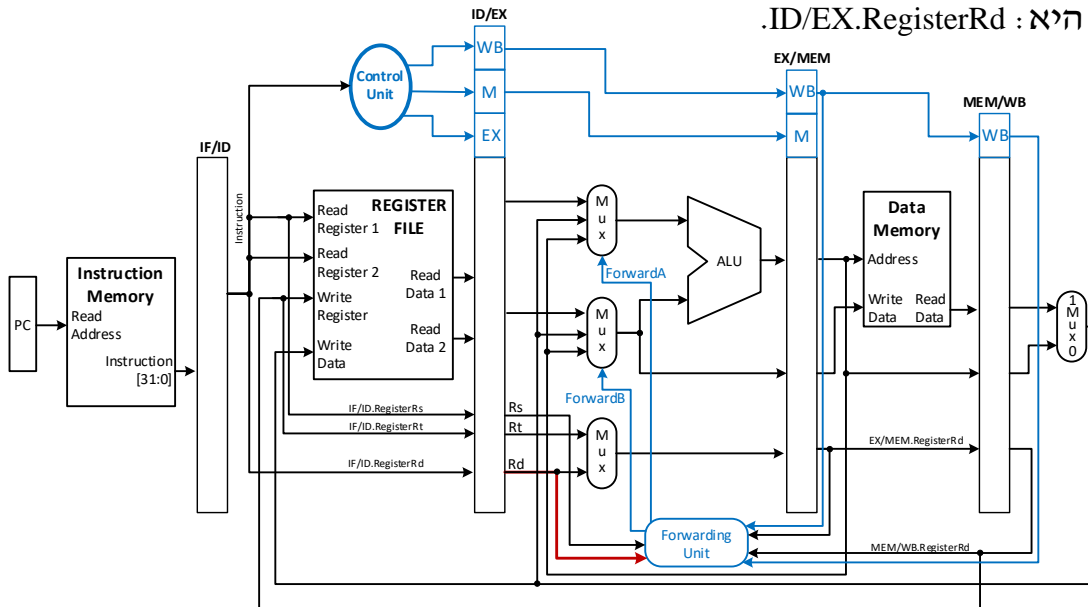
```
add $5, $6, $8
sub $3, $5, $6
sub $2, $10, $5
add $10, $5, $5
```



- א. כתבו את הפקודה שנמצאת בכל שלב משלבי הצנרת בחלק העליון של התרשים עבור פעימת השעון החמישית.
- ב. השלימו את הערכים המסומנים ב-? שעל קווי הנתונים וקווי הבקרה. הניחו כי התוכן של כל אוגר שווה למכפלת מספרו פי 100, למשל:  $5 = 500$ . במידה ולא ניתן לדעת ערך מסוים סמנו X.
- ג. עקב תקלה מסוימת, מנגנון החצייה של מקבץ האוגרים התהפך - כלומר במחזור שעון שבו נדרש לכתוב ולקרוא ממקבץ האוגרים, החומרה תבצע בפועל קריאה במחצית מחזור השעון הראשונה ולאחריו כתיבה במחצית מחזור השעון השנייה. האם הקוד יעבוד בצורה תקינה? אם כן - נמקו. אם לא - ציינו מה ייפגע במקרה זה וכיצד ניתן לתקן זאת.

3) לפניכם תרשים מעבד העובד בשיטת הצנרת וכולל את יחידת ההעברה בלבד. עקב תקלה מסוימת בחיווט הכניסות ליחידה החומרתית מאוגר הצנרת ID/EX, הכניסה R<sub>t</sub> התחלפה לה עם הכניסה R<sub>d</sub>. כלומר כעת הכניסה השנייה ליחידה

היא: ID/EX.RegisterR<sub>d</sub>.

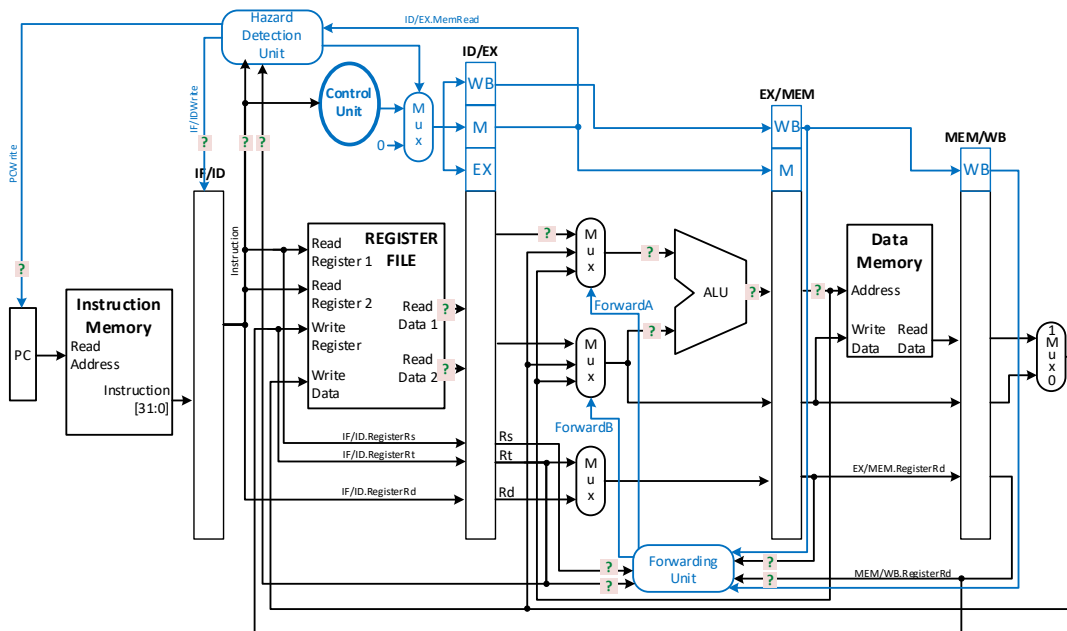


- א. האם הדבר ישפיע על קבלת ההחלטה של יחידת ההעברה לקדם מידע? אם לא - נמקו, אם כן - הסבירו.
- ב. כדי להתגבר על התקלה הוצע לקדם את סיביות Ins[20:16] מאוגר הצנרת IF/ID ליחידת ההעברה.
- (1) האם כעת פעילות המעבד תהיה תקינה? נמקו.
- (2) האם הקוד הבא ירוץ בצורה תקינה במעבד? כמה מחזורי שעון יצטרך המעבד כדי לבצע אותו?

```
add $4, $5, $6
sub $17, $4, $2
add $10, $4, $4
```



ד. במידה ויחידת איתור הסיכונים לא הייתה קיימת במעבד, האם יהיה צורך נוסף בהכנסת bubbles לקטע הקוד? אם כן אז כמה ואיפה? כתבו את הקוד המתאים למקרה זה.



### שאלות אמריקאיות כלליות עם סיכוני נתונים:

- 5) רוצים לפתור בצורה יעילה בעיה של קידום מידע במקרה של פקודת lw ואחריה sw הניגשות לאותו אוגר היעד. לשם כך שוקלים להוסיף חומרה ייעודית להעברה קדימה של ערכים. אלו ערכים יש להעביר ומאילו שלבים?
  - א. החומרה תיקח את התוכן מאוגר הצנרת MEM/WB ותקדם אותו לשלב ה-MEM. נשווה את Rt משלב ה-WB עם Rt משלב ה-MEM. החומרה תידלק רק אם התקבל חיווי על פקודת lw ואחריה sw.
  - ב. החומרה תיקח את התוכן מאוגר הצנרת MEM/WB ותקדם אותו לאוגר הצנרת EX/MEM. נשווה את Rt משלב ה-WB עם Rs משלב ה-EX. החומרה תידלק רק אם התקבל חיווי על פקודת lw ואחריה sw.
  - ג. החומרה תשווה את Rt משלב ה-WB עם Rt משלב ה-MEM. במידה והם זהים היא תכניס bubble בשלב ה-MEM ותעכב את פקודת ה-sw. החומרה תידלק רק אם התקבל חיווי על פקודת lw ואחריה sw.
  - ד. לא ניתן ליעל מקרה שכזה מכיוון ששלבי הצנרת MEM ו-WB הם אחרי שלב ה-EX המבצע את החישוב המרכזי של כל פקודה.



(6) במעבד MIPS העובד בשיטת הצנרת, הכולל את יחידת ההעברה ואת ה-HDU בלבד, איזה מידע לא נכתב לאוגר הצנרת ID/EX בסוף כל פעימת שעון :

- א. כתובות האוגרים Rs, Rt, Rd ממקבץ האוגרים.
- ב. הערך של האוגר Rs (כלומר : \$Rs).
- ג. קווי הבקרה של שלבי ה-MEM וה-WB של הפקודה הנמצאת בשלב ה-ID.
- ד. ערך כתובת הקפיצה בפקודת branch (כלומר : Branch Target).
- ה. הערך המיידי לאחר הרחבת סימן המחושב עבור הפקודה שנמצאת בשלב ה-ID.

(7) קבעו איזה משפט מבין המשפטים הבאים אינו נכון :

- א. במעבד חד מחזורי ה-CPI הוא תמיד 1.
- ב. במעבד צנרת של 5 שלבים, כל פקודה אורכת לפחות 5 מחזורי שעון.
- ג. במעבד חד מחזורי אין סיכונים מאף סוג.
- ד. משך ביצוע פקודה במעבד צנרת תמיד יהיה קצר יותר ממשך ביצועה במעבד חד מחזורי.
- ה. יחידת קידום מידע (Forwarding unit) במעבד הצנרת מסוגלת לקדם מידע משלבי ה-MEM וה-WB לשלב ה-EX בלבד.

(8) קבעו איזה משפט מבין המשפטים הבאים אינו נכון לגבי מעבד הצנרת :

- א. יחידת קידום המידע (Forwarding unit) אינה מסוגלת לקדם מידע לשלב ה-ID.
- ב. יחידת ה-HDU (Hazard Detection Unit) נועדה לטיפול רק במקרים של Load use.
- ג. יחידת קידום המידע (Forwarding unit) נועדה גם כדי לקדם מידע משלב ה-WB לשלב ה-MEM.
- ד. כתובת אוגר היעד שעובר בין אוגרי הצנרת מהאוגר ID/EX לאוגר EX/MEM תלוי בסוג הפקודה הנמצאת בשלב ה-EX.
- ה. בפקודות R-Type תעבור הכתובת של Rd ובפקודות lw ו-sw תעבור הכתובת של Rt.
- ה. יחידת ה-HDU (Hazard Detection Unit) מקבלת את כתובתו של האוגר Rt מהפקודה שנמצאת בשלב ה-EX.

(9) במודל מעבד ה-MIPS32 הכולל צנרת של 5 שלבים, ראינו כי לאוגר הצנרת ID/EX מכניסים את הערך IF/ID.RegisterRt פעמיים. מה היא המטרה של כל קו?

- א. מדובר בטעות תכנון שעברה מדור לדור ולכן נשארה כ-legacy במודל המעבד.
- ב. ערך אחד מגיע משלב ה-WB וערך שני מגיע מאוגר הצנרת הראשון, IF/ID, אך בתרשים מקובל לציין את שני הקווים ממוצא אוגר הצנרת IF/ID.
- ג. מאחר וה-MUX הנמצא בכניסה לאופרנד השני של ה-ALU, ה-ALUSrc, אינו מופיע בתרשים, מקובל לציין שני קווים : הראשון מיועד ליחידת ההעברה והשני ל-MUX.

- ד. קו אחד מיועד לציון אוגר היעד של פקודות I-Type וקו שני מיועד לציון אוגר היעד של פקודות R-Type.
- ה. לכל קו מטרה אחרת: האחד נועד ליחידת ההעברה קדימה, והשני נועד לכניסה ל-MUX של ה-RegDst וכניסה ליחידת ה-HDU.

**10) מהו ההבדל העקרוני בבניית מקבץ האוגרים (Register File) בין מעבד חד-מחזורי לבין מעבד העובד בשיטת הצנרת?**

- א. במעבד החד מחזורי ניתן לכתוב ולקרוא ממקבץ האוגרים באותו מחזור השעון, בעוד שבמעבד הצנרת יש להמתין למחזור השעון הבא על מנת לקרוא תוכן ממקבץ האוגרים.
- ב. במעבד הצנרת ניתן לכתוב למקבץ האוגרים במחצית הראשונה של מחזור השעון ולקרוא ממנו במחצית השנייה של מחזור השעון, כאשר הכתיבה היא של תוכן הפקודה הנמצאת בשלב ה-WB בצנרת. במעבד החד מחזורי, התוכן נכתב למקבץ האוגרים רק בסיום מחזור השעון.
- ג. במעבד הצנרת ניתן לכתוב למקבץ האוגרים את התוכן של הפקודה הנמצאת בשלב ה-ID במחצית הראשונה של מחזור השעון ולקרוא ממנו את תוכן זה בלבד במחצית השנייה של מחזור השעון. במעבד החד מחזורי לעומת זאת, התוכן נכתב למקבץ האוגרים רק בסיום מחזור השעון.
- ד. בשני המעבדים, כתיבה למקבץ האוגרים מתבצעת בתחילת מחזור השעון, אך במעבד הצנרת ניתן גם לקרוא תוכן במחצית השנייה של מחזור השעון.

## תשובות סופיות:

- (1) א. שני סיכונים של \$20 מפקודה 1 לפקודות 2 ול-3.  
ב. שני הסיכונים ניתן למנוע בעזרת יחידת ההעברה.  
ג. (1) לא. ג. (2) יש להכניס שני nops.
  - (2) א. להלן השלבים:  
IF:  $\langle \rangle$  ID: add \$10, \$5, \$5 EX: sub \$2, \$10, \$5  
MEM: sub \$3, \$5, \$6 WB: add \$5, \$6, \$8  
ב. ראו השלמות בסרטון הוידאו באתר.  
ג. לא.
  - (3) א. הדבר ישפיע.  
ב. (1) לא תהיה התנהגות תקינה (ראו דוגמה בסרטון הוידאו).  
ב. (2) סיכון נתונים ראשון ייפתר, אך סיכון נתונים שני לא ייפתר.  
יהיה צורך ב-7 מחזורי שעון.
  - (4) א. קיימים 4 סיכונים נתונים.  
ב. ראו תרשים מפורט בסרטון הוידאו.  
ג. יש צורך ביחידת איתור הסיכונים. ראו פירוט בסרטון הוידאו.  
ד. יש להכניס שני nops לאחר פקודת lw.
- (1) תשובה א.
  - (2) תשובה ד.
  - (3) תשובה ד.
  - (4) תשובה ג.
  - (5) תשובה ה.
  - (6) תשובה ב.

## סיכוי בקרה בצנרת:

### סיכום כללי:

#### תיאור הבעיה:

כאשר מופיעה פקודת קפיצה מותנית בתכנית, פקודות המכונה המופיעות אחריה עשויות להתברר בדיעבד כפקודות שאינן אמורות להתבצע.

נעסוק בעיקר בפקודות beq ו-bne אשר ההחלטה בהן האם לבצע קפיצה לפקודה שאינה בסדר הפקודות תתבצע רק בשלב ה-MEM. היות ו-PC מתעדכן עם כתובת הפקודה הבאה בכל מחזור שעון, לא ניתן לדעת האם הפקודות שבוצעו לאחר פקודת הקפיצה הן תקינות.

#### פתרון בסיסי:

הוספת 3 השהיות (nops) לאחר כניסתה של פקודת branch.

**זה הוא פתרון שמרני אך לא יעיל ויגרור ביצועים נמוכים של המעבד.**

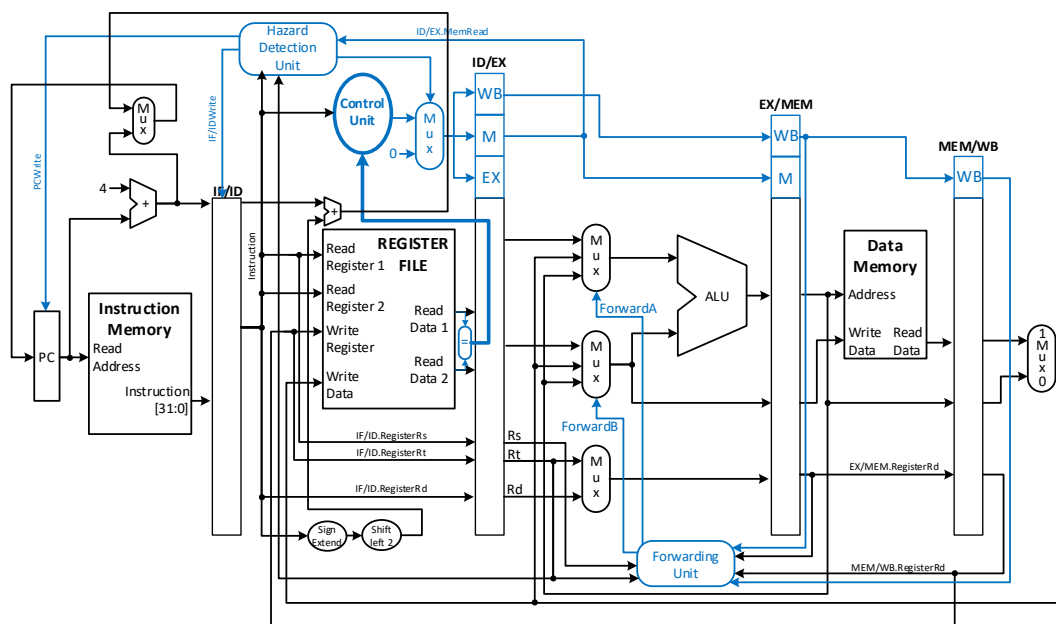
#### פתרון חומרתי:

- קידום בדיקת תנאי הקפיצה של ה-branch לשלב ה-ID.
  - חישוב כתובת הפקודה הבאה בשלב ה-IF אליה יש לקפוץ במידה והתנאי מתקיים.
  - בדיקה האם תנאי הקפיצה מתקיים (ע"י השוואת  $R_s = R_t$ ).
- שטיפת מידע (Flush) מאוגרי הצנרת.

#### הוספת nops חומרתית:

במקרה שבו קיימת פקודה מסוג R-Type ואחריה פקודת קפיצה מותנית, ופקודת ה-R-Type הכותבת לאוגר יעד אשר מתבקש לצורך בדיקת תנאי הקפיצה, נכניס nop אחד לפני פקודת הקפיצה המותנית.

## מבנה הצנרת עם קידום תנאי הבדיקה לשלב ID:



## סיכום מקומי:

נניח כי בדיקת תנאי הקפיצה מתבצעת בשלב ה-ID. כדי לייעל את פתרון הכנסת ה-nops לצנרת נבצע:

- בכל פעם שנכנסת פקודת קפיצה מותנית נחשב חומרית את ערך כתובת הקפיצה ונשווה בין שני האוגרים המהווים את תנאי הקפיצה.
- במידה וקיימת פקודת R-Type לפני פקודת הקפיצה המותנית אשר כותבת לאוגר יעד הנדרש לחישוב תנאי הקפיצה, נכניס nop בודד לפני פקודת הקפיצה.
- במידה וקיימת פקודת lw לפני פקודת הקפיצה המותנית אשר כותבת לאוגר יעד הנדרש לחישוב תנאי הקפיצה, נכניס שני nop לפני פקודת הקפיצה.
- אם בשלב השני של פקודת הקפיצה, תנאי הקפיצה מתקיים, נכניס nop אחד לאחר פקודה זו ונשטוף את תוכן אוגר הצנרת IF/ID ואת ה-PC.

## שיטות חיזוי סטטיות:

במסגרת ההצעות לייעול פעולת המעבד נבחן שני סוגים של פתרונות אפשריים:

- חיזוי סטטי (Static Branch Prediction):  
נניח תמיד כי תנאי הקפיצה מתקיים או לא מתקיים.  
זה הפתרון הפשוט ביותר והוא מתאים לצגרת קטנה בת 5 שלבים.
- חיזוי דינאמי (Dynamic Branch Prediction):  
נאפיין את ההתנהגות של כל פקודת קפיצה מותנית ונחזה את תנאי הקפיצה בזמן ריצת התכנית.  
מתאים למקרים של צגרת מורכבת יותר ומספר פקודות שרצות במחזור שגון.

## הנחות סטטיות - branch taken לעומת branch not-taken:

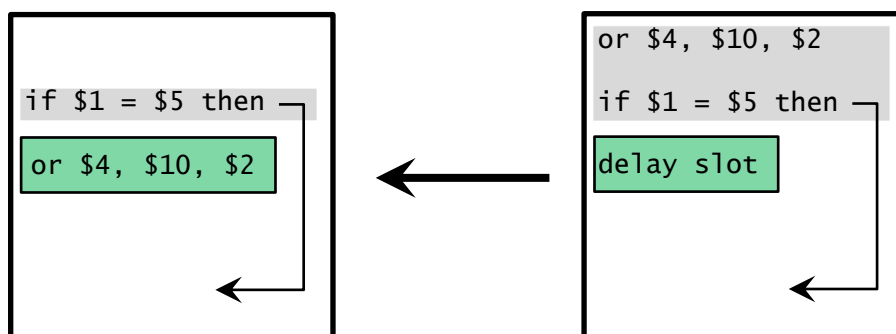
נתחיל מלבחון כיצד לייעל קוד בשיטת עיכוב פקודות branch בכך שנניח אחת משתי הנחות סטטיות לגבי פקודות אלו:

- branch taken - מניחים כי תנאי הקפיצה מתקיים.  
בפקודת beq נקפוץ לכתובת התווית ובפקודת bne לא נקפוץ לתווית.  
אלא נמשיך לפקודה הבאה.
- branch not-taken - מניחים כי תנאי הקפיצה לא מתקיים.  
בפקודת beq נמשיך לפקודה הבאה ובפקודת bne נקפוץ לתווית.

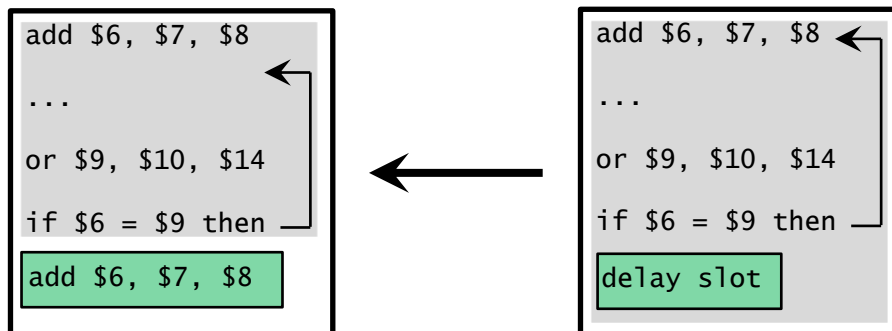
## שיטת עיכוב פקודת הקפיצה המותנית (Delayed Branch):

במסגרת שיטה זו, נשנה את סדר הפקודות בתכנית על מנת לחסוך את ה-nops שיש להכניס במצבים שתוארו לעיל. ייתכנו שלושה מקרים אפשריים:

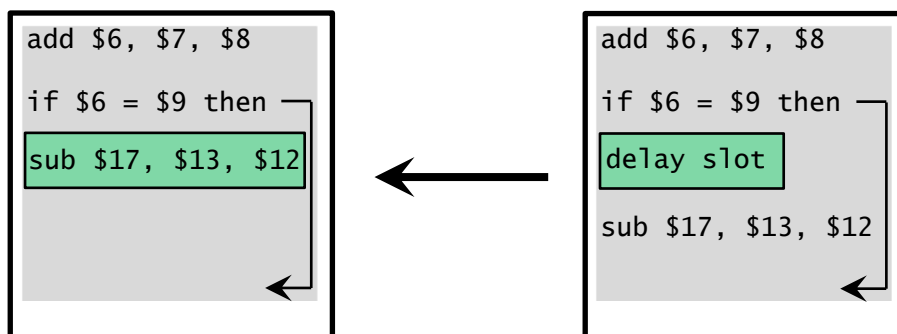
### עיכוב הפקודה from before:



**עיכוב הפקודה :from target**



**עיכוב הפקודה :from fall-through**



**תזמון פקודות (Scheduling):**

אם המהדר מתוכנן לבצע התאמה של הקוד עבור delay slots אז קיים אופטימיזר שמחפש את הפקודות המתאימות לשילוב לפי המקרים שתיארנו לעיל. לתהליך שיכתוב הקוד הנ"ל קוראים בשם scheduling.

## שיטות חיזוי דינאמיות:

### חיזוי קפיצות דינאמי (Dynamic Branch Prediction):

נרצה לאפיין כל פקודת קפיצה בשל עצמה ולהחליט על החיזוי של תנאי הקפיצה בזמן ריצת התכנית. אם פקודה מסוימת קופצת לרוב, אז נניח branch taken ואם פקודה אחרת בתכנית אינה קופצת לרוב אז נניח branch not-taken.

### חיזוי דינאמי בסיבית אחת:

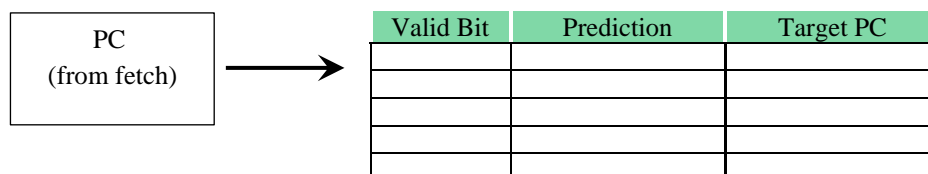
לכל 10 איטרציות נכונות החיזוי היא ב-80% מהמקרים.

### חיזוי דינאמי בשתי סיביות:

לכל 10 איטרציות נכונות החיזוי היא ב-90% מהמקרים.

### טבלת כתובות יעד של פקודות קפיצה מותנות - Branch Target Buffer (BTB):

כדי להתגבר על עניין חישוב כתובת היעד, נכיר יחידה חומרית נוספת הממוקמת בשלב ה-IF. היחידה שומרת את המידע הבא לגבי פקודות ה-branch שבתכנית מסוימת:



### הערות:

- 1) במידה וה-BTB חזתה נכון את הקפיצה, נעדכן אותה בכך שהחיזוי נכון. במידה והחיזוי היה שגוי נעדכן את ה-Prediction Bit לערך המתאים.
- 2) טבלת ה-BTB לא גדולה ולכן אינה יכולה להחזיק את כל פקודות ה-branch של תכנית מסוימת (במידה ומדובר בתכנית גדולה הכוללת מספר רב של פקודות קפיצה מותנית). לשמחתנו, פקודות קפיצה מותנית מתרכזות במקטעי זמן קטנים ולכן ניתן להיעזר בטבלה.
- 3) גם פקודות jump נכנסות ל-BTB ועבורן יש וודאות של 100% בחיזוי.



## שאלות:

### שאלות אמריקאיות כלליות עם כל סוגי הסיכונים:

- (1) למדנו כי במעבד MIPS העובד בטכנולוגית הצנרת ישנם שני קווי בקרה המגיעים לאוגר הצנרת IF/ID והם IF.Flush ו-IF/ID.Write. ההבדל בין קווים אלו הוא:
  - א. אין הבדל בין הקווים והם הוגדרו בשלבים שונים של תכנון מבנה המעבד.
  - ב. הקו IF.Flush מטפל בסיכוני בקרה והקו IF/ID.Write מטפל בסיכוני נתונים.
  - ג. הקו IF.Flush מנקה את תוכן אוגר הצנרת IF/ID ע"י הפיכת כל הערכים ל-0 והקו IF/ID.Write מונע כתיבה לאוגר הצנרת IF/ID בסוף פעימת שעון כל עוד הוא מוחזק על 0.
  - ד. הקו IF.Flush נועד לשנות את כתובת הקפיצה בפקודות Branch בעוד שהקו IF/ID.Write מאפשר כתיבה של כתובת הקפיצה המעודכנת.
  - ה. הקו IF.Flush תלוי בקו IF/ID.Write, כאשר IF/ID.Write = 0 אז מתבצעת שטיפה לאוגר הצנרת IF/ID אם הקו IF.Flush מאופשר (כלומר: IF.Flush = 1).
  
- (2) איזה מבין המשפטים מתאר נכונה את הטכניקה לפיה עובדת שיטת החיזוי הדינאמי של פקודת קפיצה מותנית (Branch)?
  - א. שיטת החיזוי הדינאמי מבצעת חישוב של כל כתובות היעד (ה-branch targets) של פקודות ה-branch בתכנית מסוימת ושומרת אותם ב-BTB כך שלא יהיה צורך לחשב אותן במהלך ביצוע כל פקודה כאשר היא נכנסת לצנרת.
  - ב. שיטת החיזוי הדינאמי הינה תוכניתית וכל מטרתה הוא לסדר את שורות הקוד של התכנית (scheduling) כך שניתן יהיה לצמצם stalls בצנרת.
  - ג. שיטת החיזוי הדינאמי חוזה את התנהגות פקודת ה-branch על סמך כתובת הקפיצה, ה-branch target במהלך ביצוע הפקודה בעודה בצנרת.
  - ד. שיטת החיזוי הדינאמי חוזה את התנהגות פקודת ה-branch שנכנסה לצנרת על סמך התנהגותה בפעם הקודמת שנכנסה לצנרת.

## תשובות סופיות:

- (1) תשובה ג.
- (2) תשובה ד.

## הערכת ביצועי מעבד:

### סיכום כללי:

#### מציאת ה-CPI הממוצע לתוכנית:

במעבד צנרת במקרה האידיאלי, בכל מחזור שעון יוצאת פקודה מהצנרת. כלומר, בהיעדר סיכונים נוכל לומר כי:  $CPI = 1$ . יחד עם זאת מכיוון שקיימים סיכונים נתונים וסיכונים בקרה, אשר כל אחד מעכב את הצנרת במידה ובשיעור שלו, יהיה עלינו להוסיף אותם לחישוב ה-CPI הממוצע:

$$CPI = 1 + \text{Data Hazard Stalls} + \text{Control Hazard Stalls}$$

### שאלות:

#### 1) מעבד MIPS עובד בצנרת בת 5 שלבים ובה הכרעת ה-branch מתבצעת בשלב 3.

ידוע כי בתכנית מסוימת אחוז הופעת פקודות ה-branch הוא 35%. המעבד עובד בשיטת חיזוי סטטי לפיו 75% פקודות beq לא נלקחות (כלומר: branch not taken). כידוע, אם מתברר כי פקודת beq אכן נלקחת (כלומר, תנאי הקפיצה מתקיים – branch taken) אז יש לבצע שטיפה (flush) לפקודות באוגרי הצנרת שנכנסו לאחר פקודת ה-beq ולהכניס את הפקודות מכתובת היעד, ה-branch target.

א. מה היא ההגדלה ב-CPI כתוצאה משטיפות אלו?  
כעת נתון גם כי שכיחות פקודות lw בתכנית טיפוסית הוא 40% ושכיחות הפקודות הצמודות ל-lw ודורשות השהיה הוא 20%.

ב. מה היא ההגדלה ב-CPI כתוצאה מסיכונים הנתונים?

ג. מה היא ההגדלה הכוללת ב-CPI כתוצאה מכל סוגי הסיכונים?

#### 2) מעבד CISC מורכב מצנרת בת 32 שלבים ובה הכרעת ה-branch מתבצעת בשלב 25.

ידוע כי תכנית טיפוסית שרצה על המעבד הנ"ל מורכבת מ-40% פקודות branch. מתוך פקודות ה-branch, נתון כי 80% מפקודות ה-beq נלקחות (branch taken). אם תנאי הקפיצה לא מתקיים (כלומר branch not taken) אז המעבד מבצע שטיפה (flush) לכל הפקודות שנכנסו לצנרת לאחר פקודת ה-beq ומכניס את הפקודות הבאות בתכנית לאחר פקודת ה-branch.

א. בהנחה ותנאי הקפיצה התקיים, כמה שלבי צנרת יש לשטוף?

ב. מה תהיה ההגדלה הממוצעת ב-CPI (ביחידות של cc/ins) כתוצאה מהשטיפות?

ג. בהנחה וניתן לקדם את שלב הכרעת ה-branch בצנרת ללא הגבלה,

לאיזה שלב יש לקדם אותו על מנת שההגדלה ב-CPI תהיה שווה

ל-75% מההגדלה כאשר ההכרעה מתבצעת בשלב ה-25?

(3) זמני הביצוע של חמשת שלבי הצנרת של מעבד MIPS הם :

Step	IF	ID	EX	MEM	WB
Time	200ps	300ps	500ps	800ps	400ps

סך התוספת ל-CPI במעבד זה כתוצאה מסיכונים (סיכונים נתונים וסיכונים בקרה) הוא  $1.1 \text{ [cc/ins]}$ . בוחנים אפשרות לפיצול שלב ה-MEM לשני שלבים הנקראים MEM1 ו-MEM2. ידוע כי במעבד זה, המונה 6 שלבים, משכי הזמן הם :

Step	MEM1	MEM2
Time	450ps	400ps

הפיצול הנ"ל עולה בתוספת של  $0.32 \text{ [cc/ins]}$  ל-CPI הממוצע הכולל. בהנחה כי הזמנים של שאר שלבי הצנרת נשארים ללא שינוי חשבו את המדדים הבאים :

- א. מדד ההאצה (SpeedUp) בין מעבד הצנרת המקורי (5 שלבים) לבין מעבד חד מחזורי עם משכי הזמן כמצוין לפני שלב הפיצול.
- ב. מדד ההאצה (SpeedUp) בין מעבד צנרת העובד ב-6 שלבים לבין מעבד הצנרת המקורי (5 שלבים).
- ג. מדד ההאצה (SpeedUp) בין מעבד צנרת העובד ב-6 שלבים לבין מעבד חד מחזורי עם משכי הזמן כמצוין לפני שלב הפיצול.

(4) זמני הביצוע של חמשת שלבי הצנרת של מעבד MIPS הם :

Step	IF	ID	EX	MEM	WB
Time	650ps	500ps	500ps	1000ps	600ps

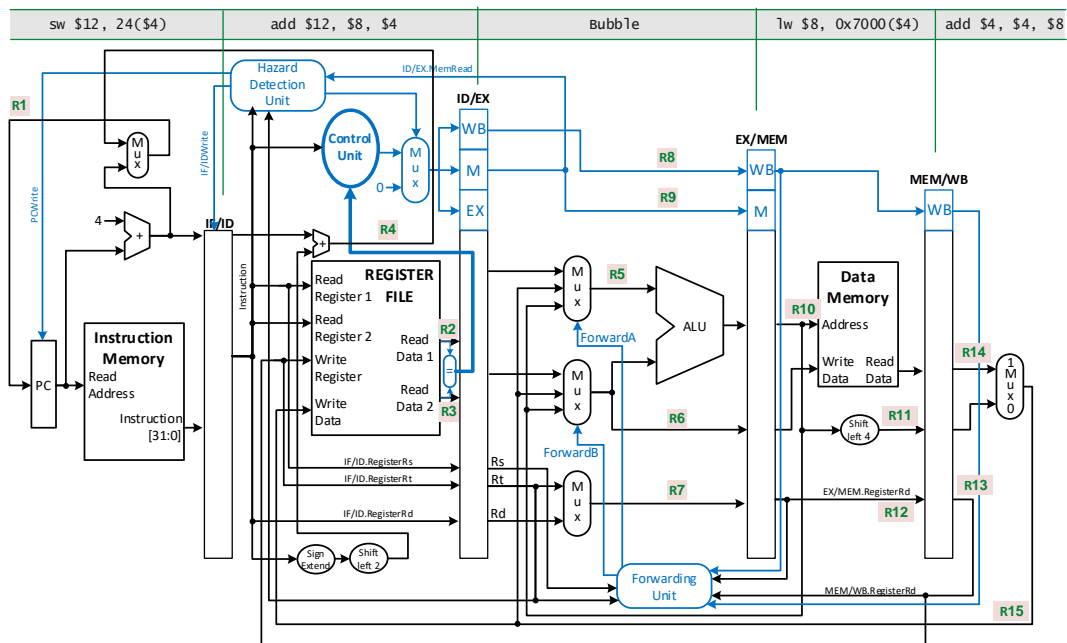
ידוע כי במעבד זה ישנה תוספת ל-CPI הממוצע הכולל של  $0.74 \text{ [cc/ins]}$  כתוצאה מסיכונים נתונים ותוספת של  $1.26 \text{ [cc/ins]}$  כתוצאה מסיכונים בקרה. מה יהיה מדד ה-SpeedUp בין מעבד זה לבין מעבד העובד באופן חד מחזורי? הניחו כי אין השהיות נוספות מעבר לאלו שצוינו.

### תשובות סופיות:

- (1) א.  $0.175 \left[ \frac{\text{cc}}{\text{ins}} \right]$  ב.  $0.08 \left[ \frac{\text{cc}}{\text{ins}} \right]$  ג.  $0.255 \left[ \frac{\text{cc}}{\text{ins}} \right]$
- (2) א. יש לשטוף 24 שלבים. ב.  $1.92 \left[ \frac{\text{cc}}{\text{ins}} \right]$  ג. יש לקדם את הכרעת ה-branch לשלב 19 בצנרת.
- (3) א.  $\text{SpeedUp} = 1.309$  ב.  $\text{SpeedUp} = 1.3885$  ג.  $\text{SpeedUp} = 1.8182$
- (4)  $\text{SpeedUp} = 1.08333$



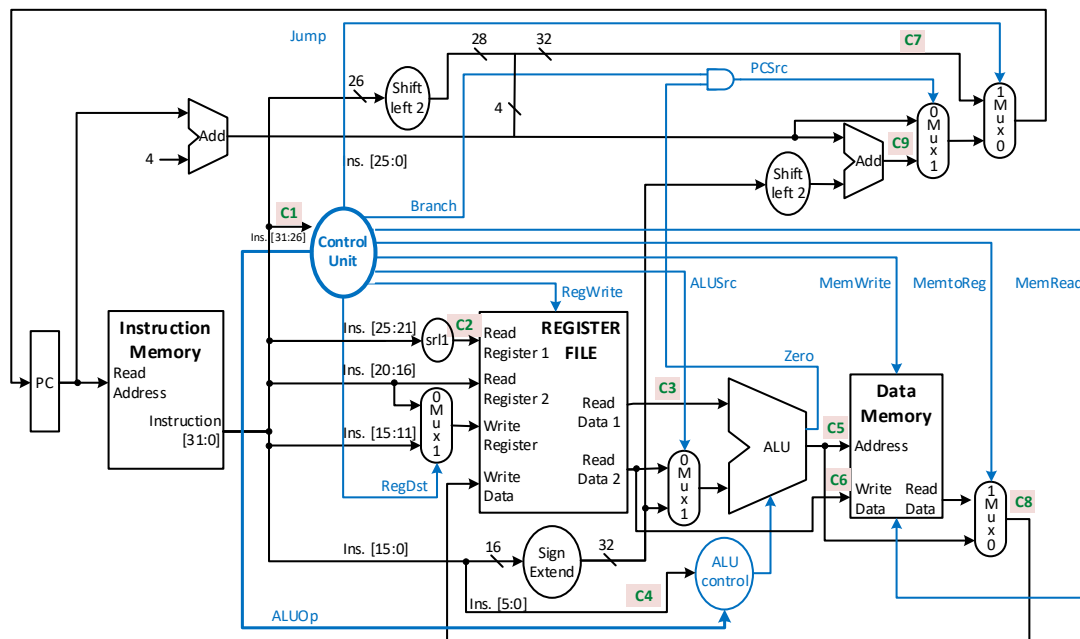
ג. בתרשים שלפניכם מתואר מעבד העובד בשיטת הצנרת ובו רכיב ה-sll4 כפי שהוגדר בתחילת השאלה. התרשים מתאר את סוף מחזור השעון השישי בביצוע הקוד הנתון (בו הפקודה: add \$4, \$4, \$8 : נמצאת בשלב ה-WB). הניחו כי סיכונים הנתונים נפתרים באמצעות יחידת העברה קדימה, יחידת איתור הסיכונים וחציית מקבץ האוגרים. מצאו את ערכי הקווים המסומנים ב-R1 – R15. נמקו את קביעתכם לכל קו וקו וצינו את החישוב שביצעתם וכן האם היה קיים סיכון נתונים ואם כן באמצעות איזו חומרה הוא נפתר. כתבו את ערכי הקווים בבסיס הקסדצימלי.



## 2) לפניכם קטע קוד בשפת אסמבלי :

Address	Code	Basic	Source
0x06009A60	0x024B2024	and \$4, \$18, \$11	1 and \$4, \$18, \$11
0x06009A64	0x21040064	addi \$4, \$8, 0x00000064	2 addi \$4, \$8, 100
0x06009A68	0x8D099A00	lw \$9, 0x00009A00(\$8)	3 lw \$9, 0x9A00(\$8)
0x06009A6C	0x01322022	sub \$4, \$9, \$18	4 sub \$4, \$9, \$18
0x06009A70	0xAC84002C	sw \$4, 0x0000002C(\$4)	5 sw \$4, 44(\$4)

ערכו של כל אוגר לפני ביצוע הקוד מאותחל למספרו כפול 0xAB0000. כלומר, הערך השמור באוגר 1 הוא 0xAB0000, הערך השמור באוגר 2 הוא 0x1560000 וכן הלאה. ניתן לגשת לכל מרחב הזיכרון בפקודות הגישה לזיכרון (sw ו-lw) כל עוד הכתובת מתחלקת ב-4. בתרשים המעבד החד-מחזורי שמופיע למטה, נוסף רכיב srl1 המבצע הזזה ימינה במקום אחד של ערך המתקבל בכניסתו. הרכיב ממוקם בכניסת Read Register 1 של מקבץ האוגרים ועובד על ערך הסיביות [25:21].Ins.



א. (1) כתבו את ערכם ההתחלתי השמור באוגרים 9 ו-11 כמילה ב-8 ספרות הקסדצימליות.

(2) פעולת srl1 מתאימה לפקודת האסמבלי הבאה : srl \$Rt, \$Rs, 1.

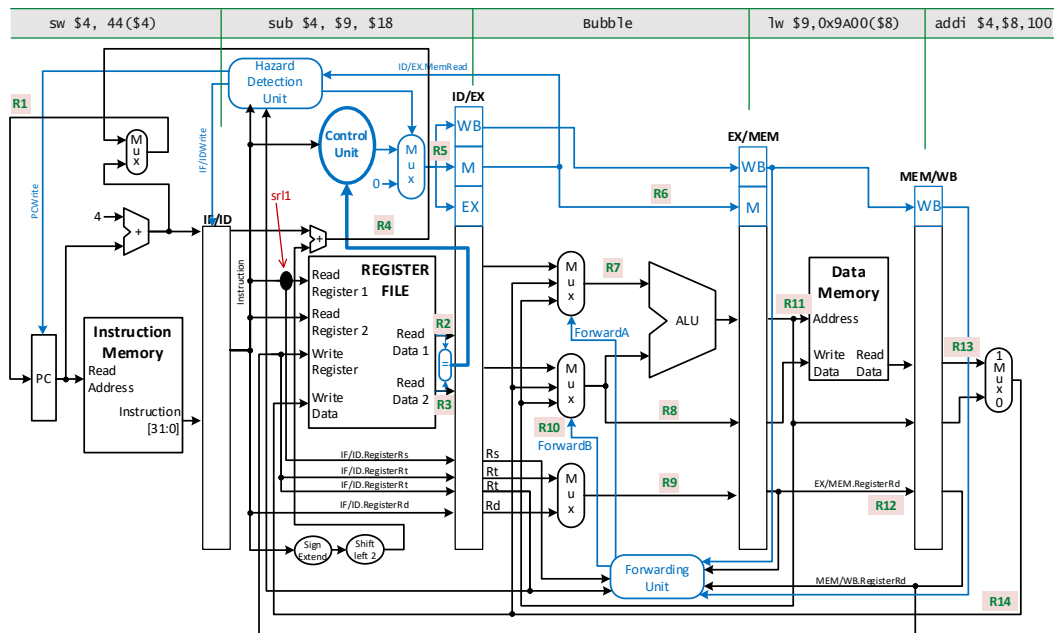
במה שונה תוצאת פקודה זו מהפקודה : sra \$Rt, \$Rs, 1?

לצורך ההסבר תוכלו להיעזר בקוד הבא (חשבו מה נכתב לאוגרים t1 ו-t2) :

```
li $t0, -35      # Initializing $t0 = -35
sra $t1, $t0, 1
srl $t2, $t0, 1
```

ב. חשבו את ערכי הפרמטרים C1-C9 על הקווים המופיעים בתרשים בהנחה שהמידע נבדק לקראת סוף פעימת השעון בביצוע הפקודה השלישית בקוד: `lw $9, 0x9A00($8)`. הניחו כי ערכי קווי הבקרה הם כפי שלמדנו וכתבו את התוצאות בבסיס הקסדצימלי.

ג. בתרשים שלפניכם מתואר מעבד העובד בשיטת הצנרת ובו רכיב ה-`srl1` כפי שהוגדר בתחילת השאלה ומסומן באליפסה שחורה וחץ אדום מעליה. התרשים מתאר את סוף מחזור השעון השישי בביצוע הקוד הנתון (בו הפקודה: `addi $4, $8, 100` נמצאת בשלב ה-WB). הניחו כי סיכוני הנתונים נפתרים באמצעות יחידת העברה קדימה, יחידת איתור הסיכונים וחציית מקבץ האוגרים. מצאו את ערכי הקווים המסומנים ב-R1-R14. נמקו את קביעתכם לכל קו וקו וצינו את החישוב שביצעתם וכן האם היה קיים סיכון נתונים ואם כן באמצעות איזו חומרה הוא נפתר. כתבו את ערכי הקווים בבסיס הקסדצימלי.



## תשובות סופיות:

- (1) א. (1)  $\$4 = 0xFFFFFFFF80$  ,  $\$12 = 0xFFFFFE80$  , א. (2) הכפלה פי  $2^4$  .  
 ב.  $C1 = 0xE$  ,  $C2 = 0x0$  ,  $C3 = 0xFFFF7000$  ,  $C4 = 0xB404B4DC$  ,  
 $C5 = 0xB402F4DC$  ,  $C6 = 0xFFFFFFFF00$  ,  $C7 = 0xFFFE0000$  ,  
 $C8 = 0XXXXXXXXXX$   
 ג.  $R1 = 0xB402F4E4$  ,  $R2 = 0xFFFFFFFF00$  ,  $R3 = 0xFFFF7000$  ,  $R4 = 0xB4047560$  ,  
 $R5 = 0xFFFE0000$  ,  $R6 = 0xFFFF7000$  ,  $R7 = 0x8$  ,  $R8 = 0x0$  ,  $R9 = 0x0$  ,  
 $R10 = 0xFFFE0000$  ,  $R11 = 0xFFFE0000$  ,  $R12 = 0x8$  ,  $R13 = 0x4$  ,  
 $R14 = 0XXXXXXXXXX$  ,  $R15 = 0xFFFF7000$
- (2) א. (1)  $\$9 = 0x06030000$  ,  $\$11 = 0x07590000$  , א. (2) פקודת sra מרפדת לפי סימן (Sign Extend) בעוד שפקודת srl תמיד תוסיף אפסים ללא קשר לסימן המספר.  
 ב.  $C1 = 0x23$  ,  $C2 = 0x4$  ,  $C3 = 0x06010064$  ,  $C4 = 0x0$  ,  $C5 = 0x06009A64$  ,  
 $C6 = 0x06030000$  ,  $C7 = 0x04266800$  ,  $C8 = 0x21040064$  ,  $C9 = 0x05FF026C$  ,  
 ג.  $R1 = 0x06009A74$  ,  $R2 = 0x06010064$  ,  $R3 = 0x0C060000$  ,  $R4 = 0x06011AF8$  ,  
 $R5 = 0x182$  ,  $R6 = 0x0$  ,  $R7 = 0x06009A74$  ,  $R8 = 0x0C060000$  ,  $R9 = 0x4$  ,  
 $R10 = 0x0$  ,  $R11 = 0x06009A74$  ,  $R12 = 0x9$  ,  $R13 = 0XXXXXXXXXX$  ,  
 $R14 = 0x06010064$