תוכן העניינים:

מבנה המחשב ותכן לוגי 2

מעבד העובד בשיטת הצנרת 2

מבוא לצנרת : 2

סיכום כללי : 2

מבנה הצנרת במעבד החד-מחזורי : 5

סיכום כללי : 5

שאלות : 9

תשובות סופיות : 13

סיכונים בצנרת : 14

סיכום כללי : 14

סיכוני נתונים בצנרת : 15

סיכום כללי : 15

שאלות : 21

תשובות סופיות : 25

סיכוני בקרה בצנרת : 26

סיכום כללי : 26

מבנה המחשב ותכן לוגי

מעבד העובד בשיטת הצנרת

מבוא לצנרת:

סיכום כללי:

הגדרה:

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

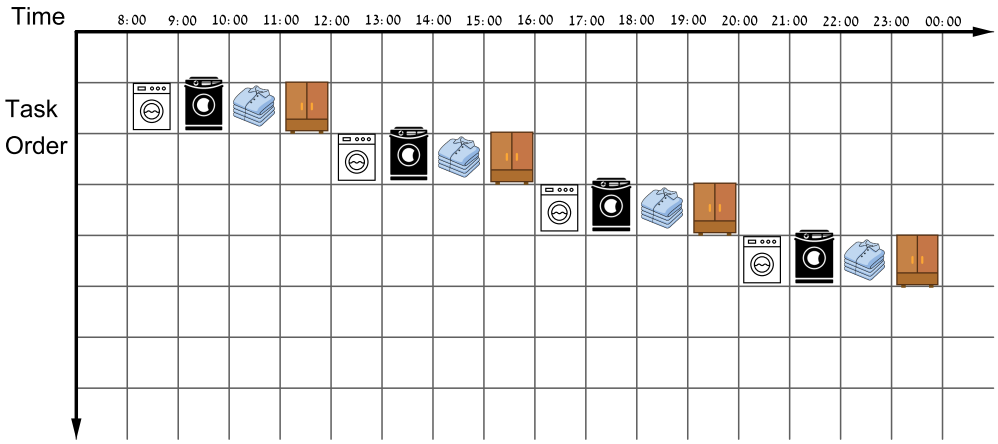
אנלוגיה שתלווה אותנו לאורך הלמידה על מהות ההצנרה ותכונותיה:

נתייחס משימת כביסה הכוללת 4 שלבים:

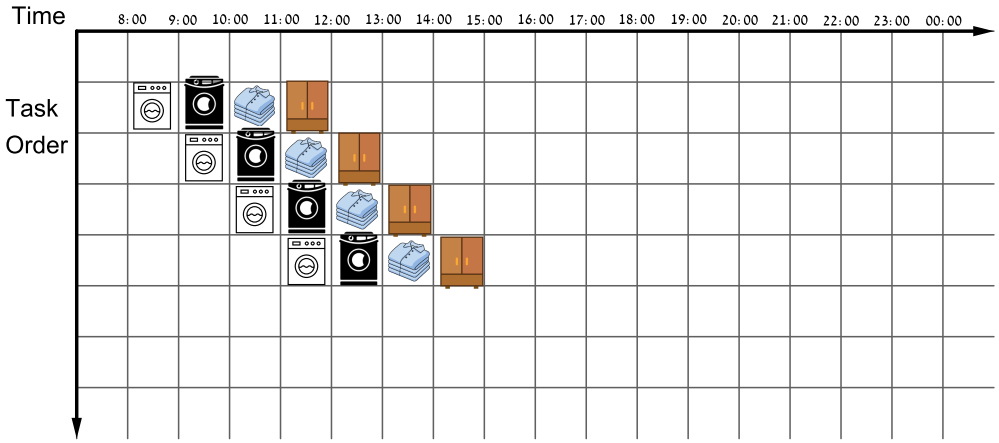
בהינתן כביסה מלוכלכת (קרי: משימת כביסה).

- 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 קָטֵן.

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:

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

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:

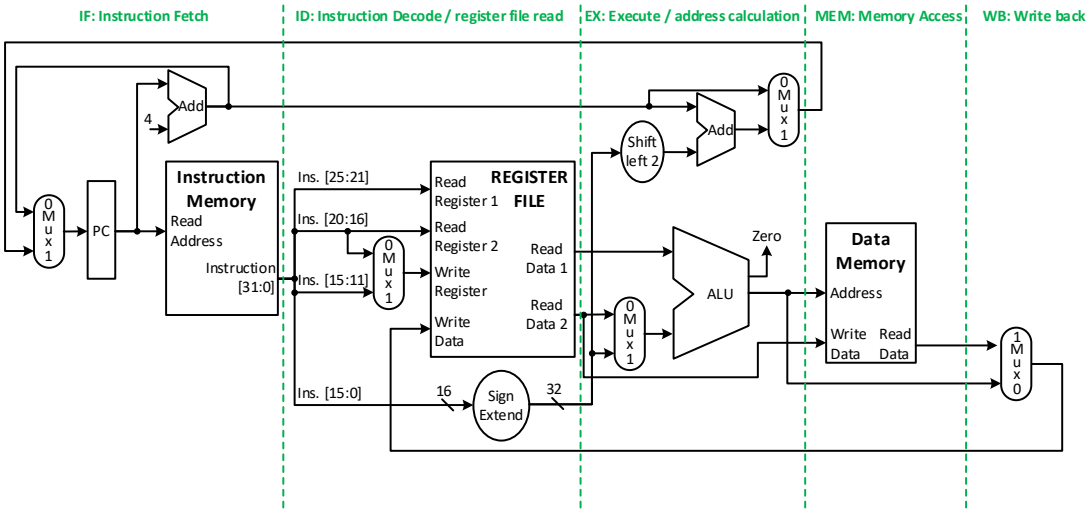
לאחר שרכשנו הבנה בסיסית לגבי תהליך ההצנרה, נוכל למנות 5 שיקולים בתכנון ה-ISA של מעבדי ה-MIPS אשר מכוונים למימוש פקודות בצורה מקבילית:

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:

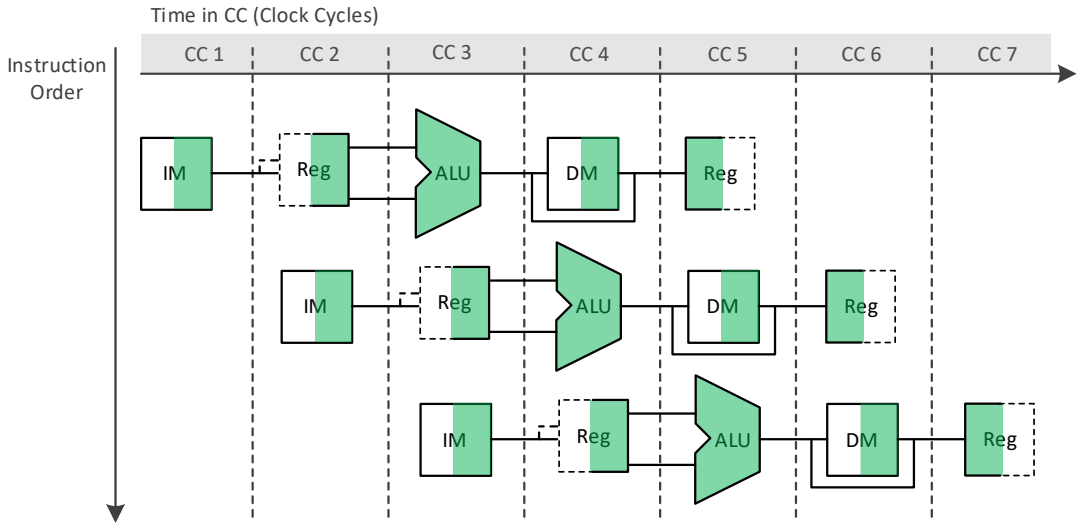


עיקרון תהליך הצנרת:

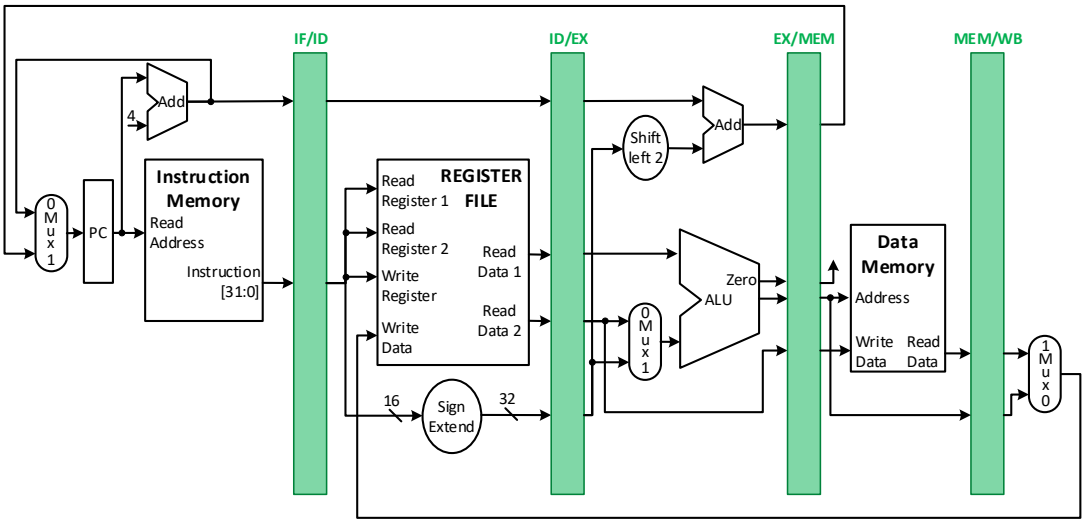
מידע זורם משמאל לימין ולא חוזר אחורנית.
יחד עם זאת יש 2 מקרים יוצאים-מן-הכלל:

1) שלב ה-WB אשר מתקיים בפקודות מסוג R-Type ובפקודת 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):



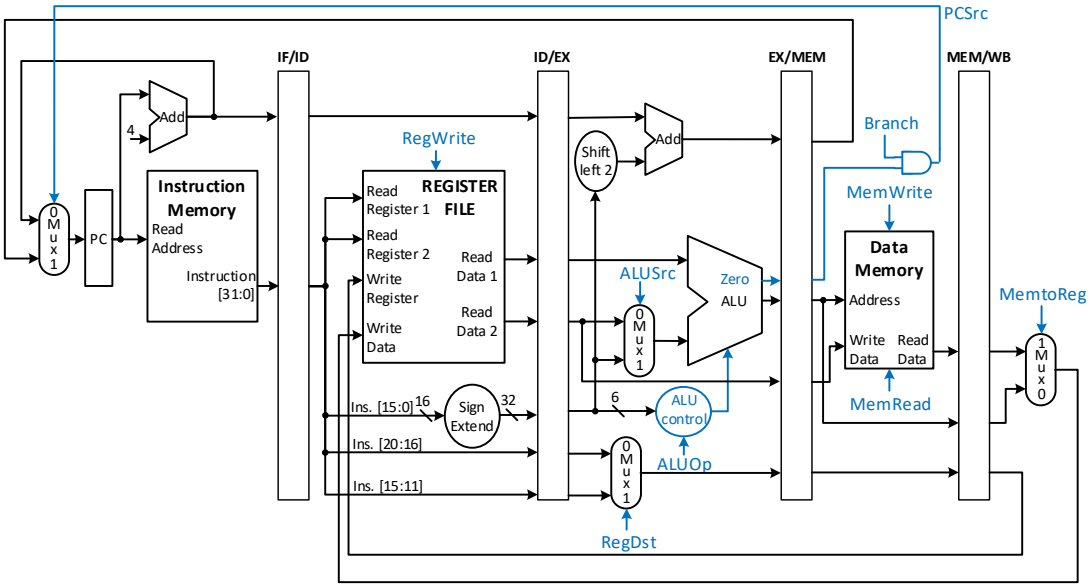
האוגרים של הצנרת נקראים לפי שני השלבים שהם מפרידים ביניהם:

- 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 שעון מסוים למחזור שעון 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 השעון 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 השעון השני מאוגר הצנרת IF/ID דרך ID/EX ויסיים באוגר הצנרת EX/MEM.
- בפקודת beq הערך של PC + 4 נכתב במחזור השעון הראשון אך יש לעשות בו שימוש רק במחזור השעון השלישי. לכן ערך זה יעבור מאוגר הצנרת IF/ID לאוגר ID/EX.

קווי הבקרה של הצנרת:



חלוקת קווי בקרה:

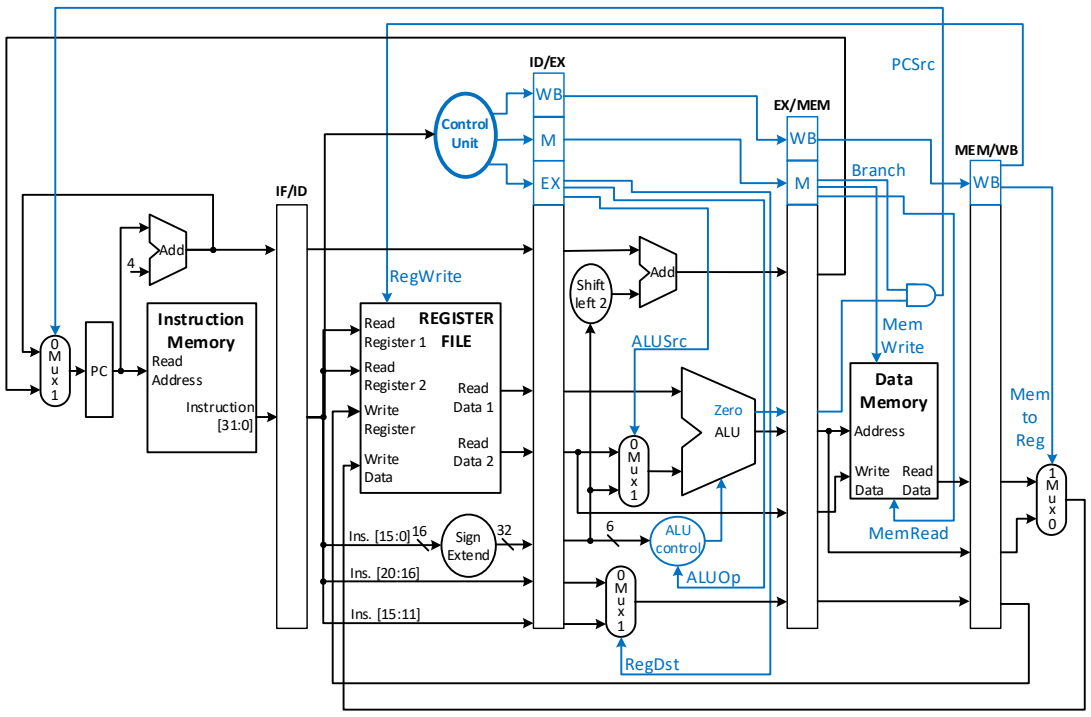
היות וכל אוגר צנרת נכתב בכל מחזור שעון, נוכל לחלק את הכתיבה של קווי הבקרה לחמשת שלבי הצנרת באופן הבא:

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

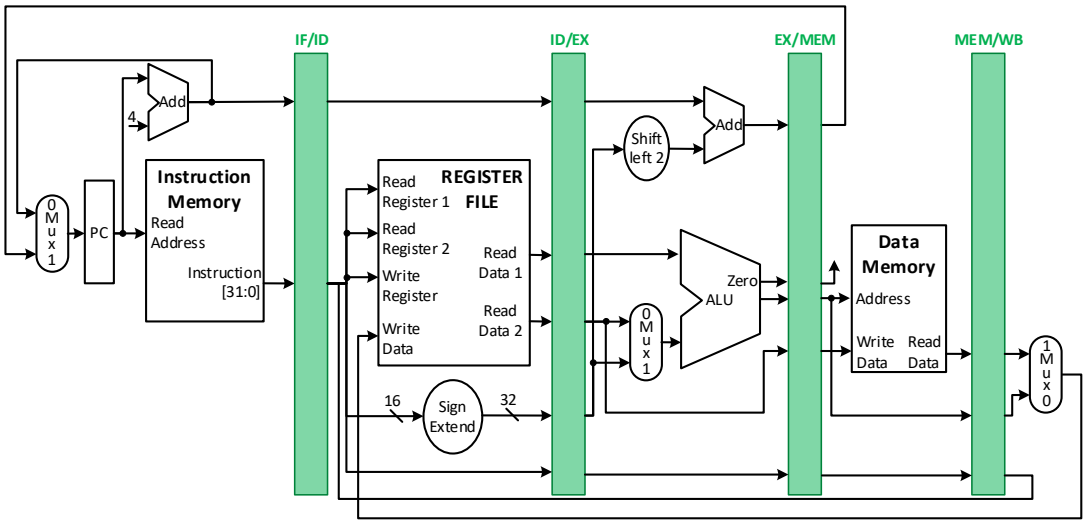
יא. ערכי קווי הבקרה עוברים דרך אוגרי הצנרת מכיוון שהם מכתיבים את הפעילות שיש לבצע עם מידע הנמצא באוגרים אלו.

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)
```

א. כמה מחזורי שעון יצטרך מעבד העובד הצנרת כדי לבצע את התכנית?

ב. 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 5-3. 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

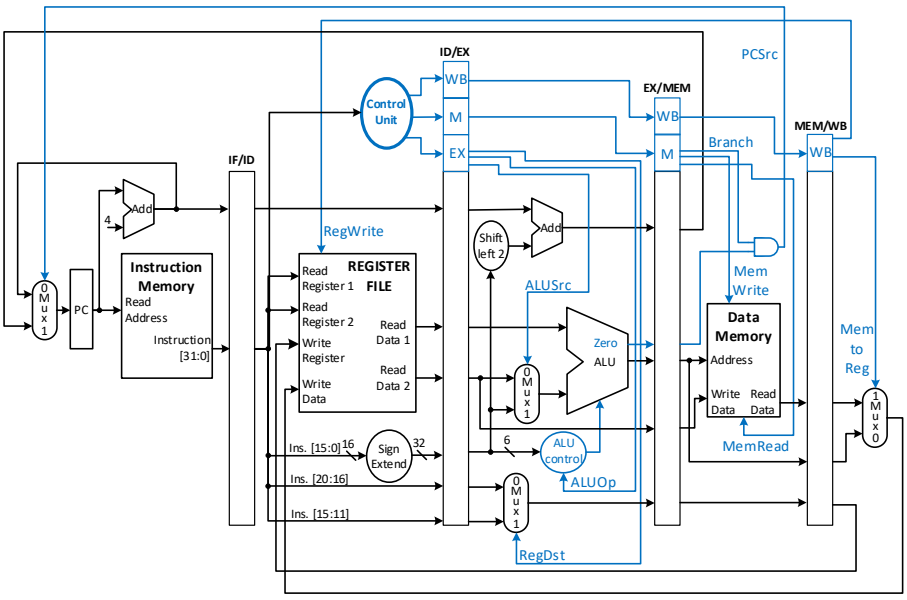
ג. שלב EX: add $4, $5, $6 שלב MEM: sw $8, 92($10) שלב WB: lw $7, 400($20)

ד. שלב EX: sub $4, $5, $6 שלב MEM: sw $8, 0($18) שלב WB: beq $7, $10, 0x49

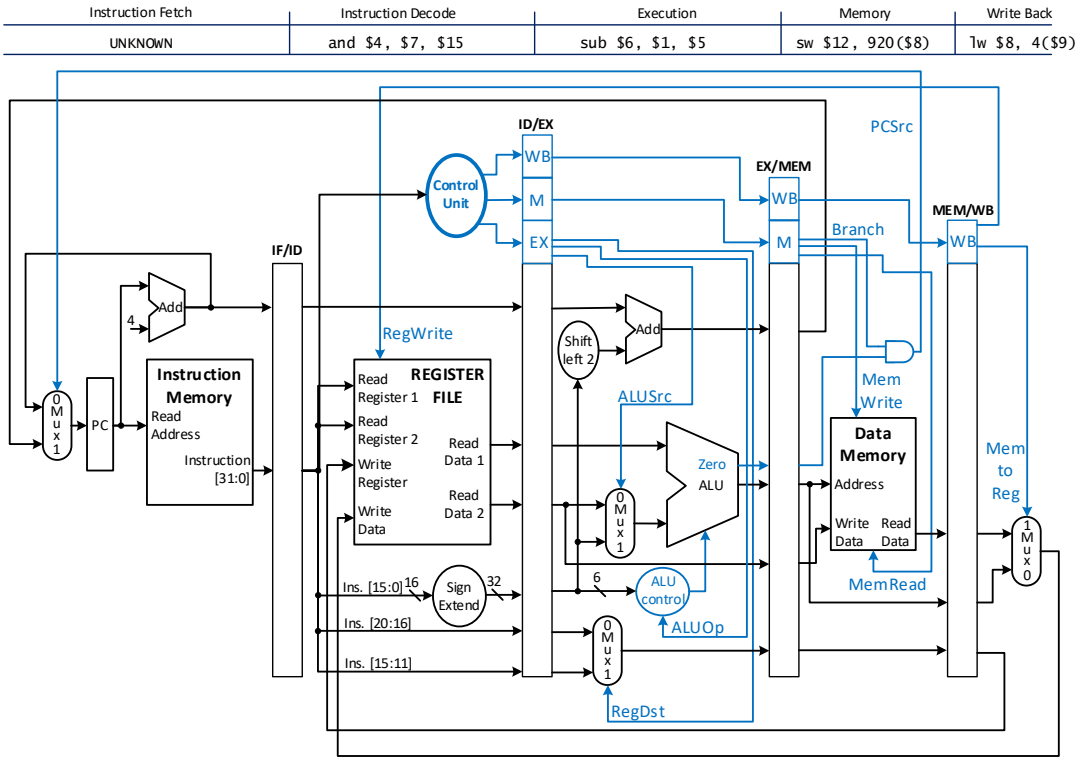
טבלת עזר:

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

ב. מהו זמן מחזור השעון לפי המודל החד-מחזורי ולפי מודל הצנרת?

ג. כמה זמן ייקח לבצע את פקודת lw לפי כל מודל?

ד. 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?

ה. 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:

IF: add $20, $16, $12 ID: lw $9, 40($7) EX: lw $8, 40($7)
MEM: ⟨ ⟩ WB: ⟨ ⟩

מחזור שעון חמישי:

IF: sub $31, $8, $6 ID: add $22, $17, $13 EX: add $20, $16, $12
MEM: lw $9, 40($7) WB: lw $8, 40($7)

מחזור שעון שביעי:

IF: ⟨ ⟩ ID: sw $10, -100($7) EX: sub $31, $8, $6
MEM: add $22, $17, $13($7) WB: add $20, $16, $12

3) תשובה ג.

4) א. EX = 1100 , MEM = 010 , WB = 11
ב. חד מחזורי: $CCT = 2050\,ps$, צנרת: $CCT = 750\,ps$.
ג. חד מחזורי: $T(\text{lw}) = 2050\,ps$, צנרת: $T(\text{lw}) = 3750\,ps$.
ד. נפצל את שלב ה-MEM ל-3 חלקים שאורך כל אחד הוא 250ps.
חד מחזורי: $T(\text{lw}) = 2050\,ps$, צנרת: $T(\text{lw}) = 3150\,ps$.
ה. התפוקה תגדל מכיוון שכעת כל פעולה מתבעצת 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 המִיקְבּוּל 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.

סיכוני 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:

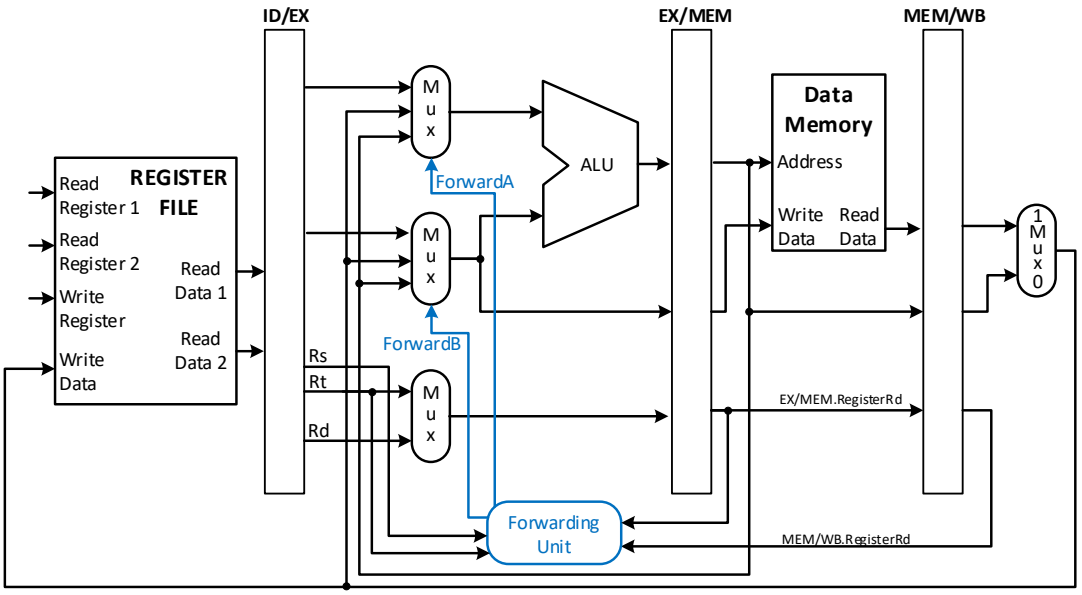
נאמץ את הסימון הבא: "ID/EX.RegisterRs" כדי לציין את מספר האוגר שהערך שלו נמצא באוגר הצנרת ID/EX והוא מיועד להיכנס כהאופרנד הראשון ליחידת ה-ALU.
באותו אופן:

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 סיכוני 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:

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

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

- סיכון נתונים בשלב ה-EX:

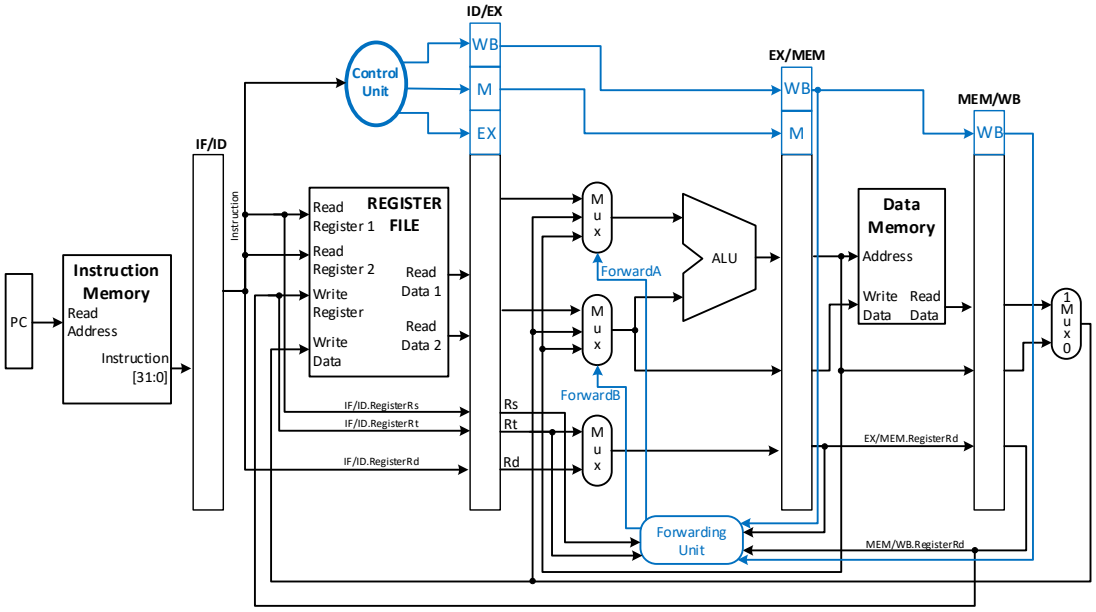
```
if      ( (EX/MEM.RegWrite)
          and (EX/MEM.RegisterRd ≠ 0)
          and (EX/MEM.RegisterRd = ID/EX.ResigterRs) )
then    ForwardA = 10

if      ( (EX/MEM.RegWrite)
          and (EX/MEM.RegisterRd ≠ 0)
          and (EX/MEM.RegisterRd = ID/EX.ResigterRt) )
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.ResigterRs) )
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.ResigterRt) )
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:

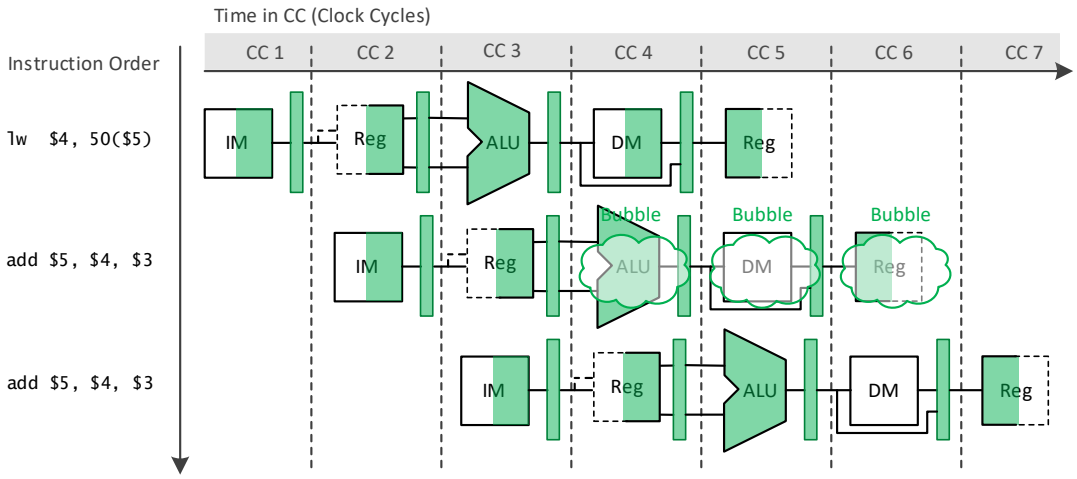
כאשר פקודת lw מופיעה לפני פקודה שמבקשת את המידע המובא מהזיכרון, לא ניתן לבצע את שתיהן ללא כל התערבות. זאת כיוון שהנתון המובא מהזיכרון לאוגרי הצנרת בפקודת ה-lw מתבצע רק בסוף מחזור השעון הרביעי לפקודה בעוד שהקריאה לנתון בפקודה הבאה מתבצעת המחזור השעון השני. לכן קריאה לנתון ע"י הפקודה המאוחרת לא אפשרית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

יחידת איתור סיכונים - Hazard Detection Unit (HDU):

נמקם HDU בשלב ה-ID בו נחליט אם להכניס עיכוב (stall) לצנרת 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 (nop = no operation או do nothing).



חציית מקבץ האוגרים:

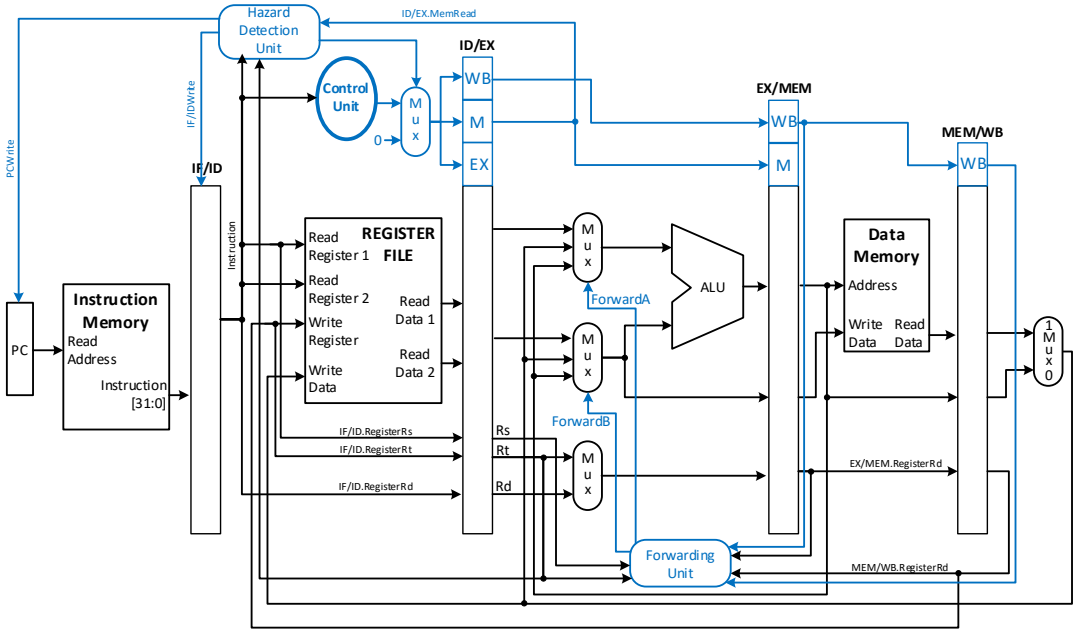
במידה ובשאלה מצוין כי ניתן לכתוב ולקרוא ממקבץ באוגרים באותו מחזור שעון (כגון שמקבץ האוגרים בנוי חומרתית בדרך המאפשרת כתיבה אליו במחצית מחזור שעון הראשון קריאה ממנו במחצית מחזור השעון שני) אז נאמר כי במקבץ האוגרים קיימת חומרה האחראית על **חציית מקבץ האוגרים**.

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:

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.

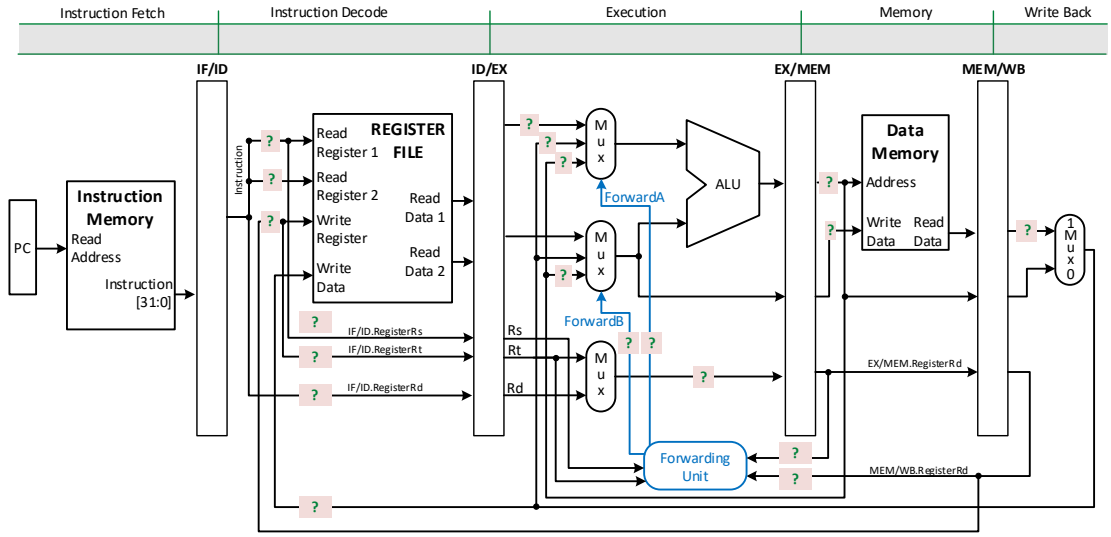
ב. אלו סיכונים ניתן למנוע בעזרת יחידת ההעברה?
ציינו את ביצוע קידום המידע הנדרש בכל מקרה.

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
```

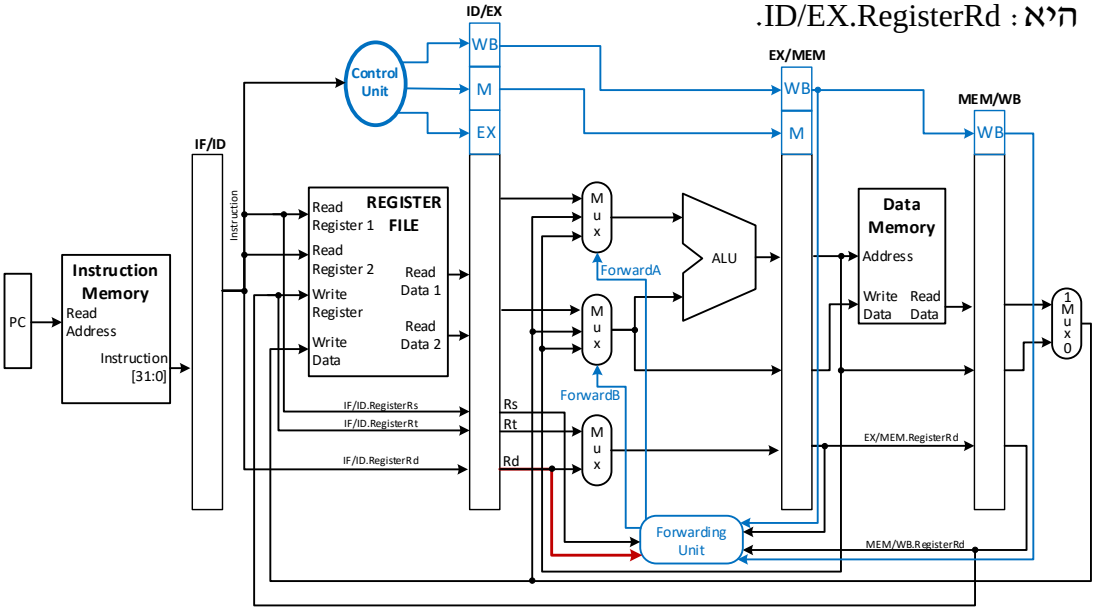


א. כתבו את הפקודה שנמצאת בכל שלב משלבי הצנרת בחלק העליון של התרשים עבור פעימת השעון החמישית.

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 Rt התחלפה לה עם הכניסה Rd. כלומר כעת הכניסה השנייה ליחידה היא: ID/EX.RegisterRd.



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?

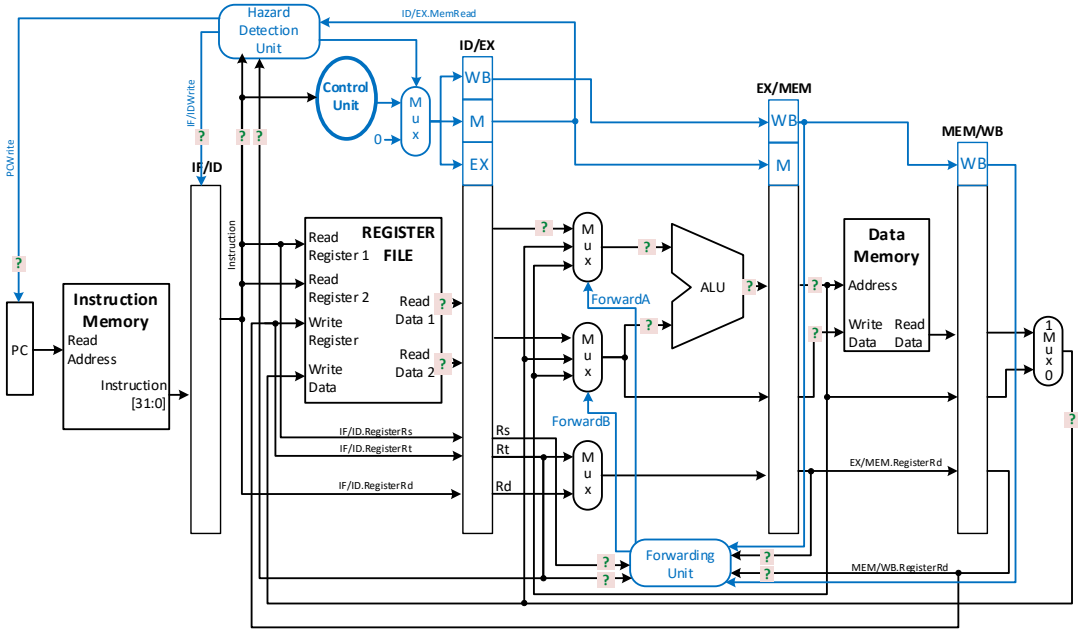
```
add $4,  $5, $6

sub $17, $4, $2

add $10, $4, $4
```

4) לפניכם קטע הקוד הבא המיושם במעבד צנרת 5 שלבים. המעבד כולל יחידת העברה (Forwarding Unit), יחידת איתור סיכונים (HDU) וחצייה של מקבץ האוגרים (כתיבה וקריאה באותו מחזור שעון ובסדר זה).

```
add $6, $7, $18

sub $6, $6, $14

lw  $7, 200($6)

add $14, $7, $6
```

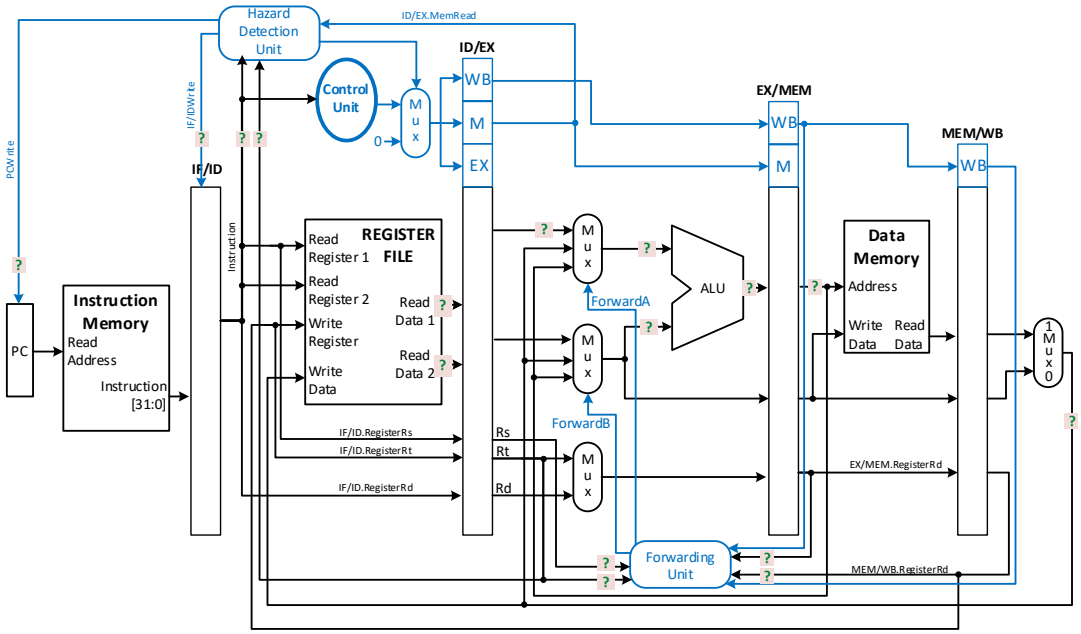
א. כמה סיכוני נתונים קיימים בקוד? בין אלו שלבים של הצנרת ובין אלו פקודות?

ב. התרשים הבא מתאר את פעימת השעון החמישית.
מלאו את הערכים בקווי הנתונים וקווי הבקרה המסומנים ב-?
לשם כך הניחו כי ערכם ההתחלתי של האוגרים שווה למספרם כפול 20.
כלומר: $1 = 20 , $2 = 40 , $3 = 60 , … .
במידה ולא ניתן לדעת מה ערכו של קו מסוים, כתבו X.
יש לכתוב את הערכים בבסיס עשרוני בלבד.



ג. האם ניתן לפתור את כל סיכוני הנתונים בעזרת יחידת ההעברה בלבד או שיש צורך גם ביחידת איתור הסיכונים? אם כן אז היכן?

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.



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: 〈 〉 ID: add $10, $5, $5 EX: sub $2, $10, $5

MEM: sub $3, $5, $6 WB: add $5, $6, $8

ב. ראו השלמות בסרטון הוידא ובאתר.
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 סיכוני 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.

סיכוני 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 וה-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.

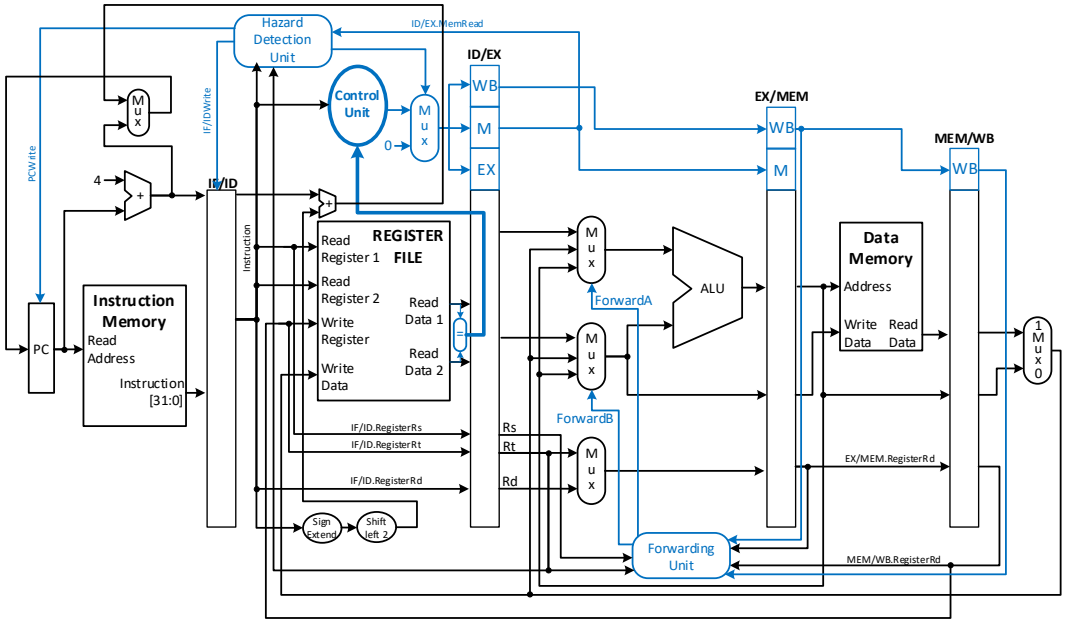
פתרון חומרתי:

- 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 Rs = Rt).
- 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 חומרתית 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 לצנרת 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 צנרת מורכבת יותר ומספר פקודות שרצות במחזור שעון.

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:

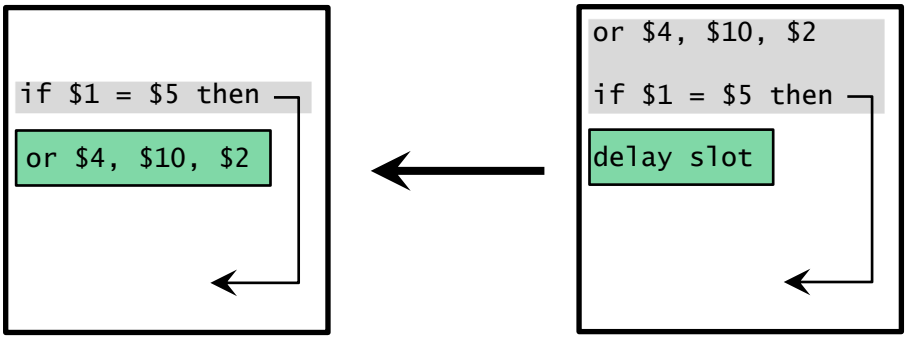
- branch taken - מניחים כי תנאי הקפיצה מתקיים.
  בפקודת beq נקפוץ לכתובת התווית ובפקודה bne לא נקפוץ לתווית אלא נמשיך לפקודה הבאה.

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

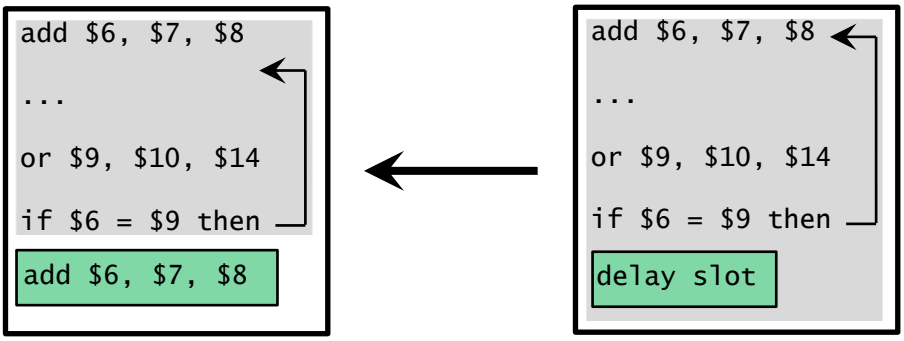
שיטת עיכוב פקודת הקפיצה המותנית (Delayed Branch):

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

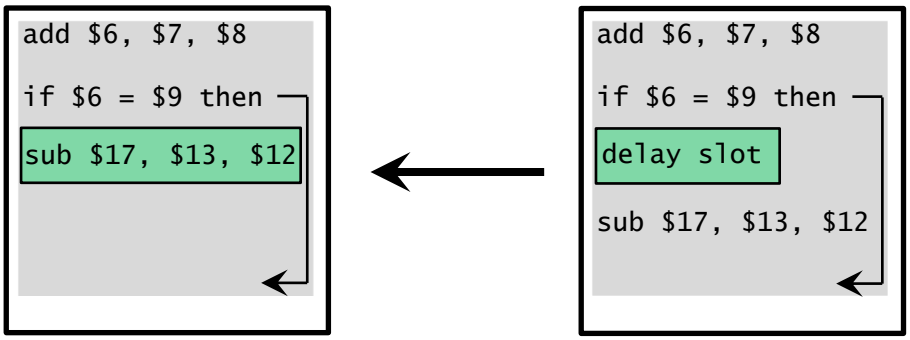
עיכוב הפקודה from before:



עיכוב הפקודה from target:



עיכוב הפקודה from fall-through:



תזמון פקודות (Scheduling):

אם המהדר מתוכנן לבצע התאמה של הקוד עבור delay slots אז קיים אופטימייזר 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 עבורה branch not-taken.

חיזוי דינאמי בסיבית אחת:

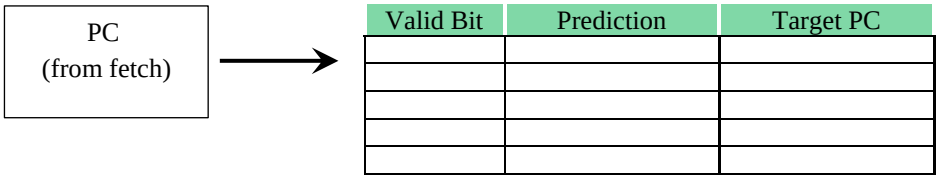
לכל 10 איטרציות נכונות החיזוי היא ב-80% מהמקרים.

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 חומרתית 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.