אוניברסיטת בר-אילן

הפקולטה להנדסה

BAR-ILAN UNIVERSITY

Engineering Faculty

מבנה מחשבים ספרתיים 83-301

תשע"ה סמס' ב' מועד ב'

שם הקורס	מבנה מחשבים ספרתיים		
מספר הקורס	83-301		
שם המרצה	פרופ' שמואל וימר + מר עמית מנדלבאום		
	תשע"ה	סמסטר ב'	מועד ב'
משך הבחינה	שלש שעות		
חומר עזר	שני ספר הקורס של המחברים Patterson & Hennessy + שקפי הקורס והתרגול בלבד (כולל הערות על גביהם). כל חומר אחר, כולל תרגילים, אמצעים אלקטרונים, מחשבונים, וכו', אסורים בשימוש. **יש לצרף את שאלוני הבחינה למחברת.**		
	יש לענות על כל השאלות. כל תשובה יש לנמק ולהסביר. כל תשובה מספרית מחייבת את הצגת דרך החישוב. סה"כ הנקודות האפשריות 110. הציון המרבי האפשרי הינו 100. **יש לכתוב בעט בלבד. כתיבה בעפרון לא תיבדק.**		

בהצלחה!

<u>שאלה א'</u> (30 נק')

נתונה מערכת זיכרון וירטואלי (VM) המתרגמת כתובות וירטואליות לכתובות פיזיות. בשאלות שלפניכם משנים פרמטר **<u>אחד בלבד</u>** במערכת. יש להסביר **<u>במדויק ובפרוטרוט</u>** מה יקרה בכל אחד מהמקרים הבאים. בכל אחת מהשאלות שלהלן, במדה ואין שנוי, הסבירו מדוע, ובמדה וישנו שינוי, ציינו את גדלו והסבירו מדוע. (1-5 3 נק', 6-8 5 נק').

1. מה יקרה למספר הסיביות בכל כניסה של טבלת הדפים (PAGE TABLE) אם גדלו של הזכרון הפיזי (בבתים) קטן פי שניים?

תשובה: מספר הסיביות יקטן ב אחד. הורדת הזכרון הפיזי לחצי פירושה שמספר הדפים קטן לחצי, ועל כן בכדי לציין את מספרו של הדף הפיזי נדרשת סיבית אחת פחות.

2. מה יקרה למספר הכניסות בטבלת הדפים אם גדלו של הזיכרון הפיזי (בבתים) קטן פי ארבע?
תשובה: מספר הכניסות לא ישתנה. הורדת הזכרון הפיזי איננה משנה את מספר הדפים הנקבע ע"י גודל הזיכרון הוירטואלי וגודל הדף.

3. מה יקרה למספר הסיביות בכל כניסה של טבלת הדפים במדה ומערכת הזכרון הוירטואלי גדלה מ 32 ל 64 סיביות?
תשובה: מספר הסיביות ישאר ללא שינוי משום שגודל כל כניסה בטבלת הדפים מוגדרת ע"י גודל של כתובת פיזית, ולא ע"י גודל הזיכרון הוירטואלי.

4. מה יקרה למספר הכניסות בטבלת הדפים במדה וגדלו של הזכרון הוירטואלי (בבתים) קטן פי שמונה?
תשובה: מספר הכניסות יקטן פי שמונה גם כן, משום שבטבלת הדפים יש כניסה לכל דף ומספרם קטן פי שמונה.

5. מה יקרה למספר הכניסות בטבלת הדפים במדה וגדלו של דף (בבתים) יקטן פי שניים?
תשובה: מספר הכניסות יוכפל, משום שבטבלת הדפים יש כניסה לכל דף ומספרם מוכפל.

6. מוסיפים למערכת TLB בעל 64 כניסות. מה קורה לגדלה של טבלת הדפים?
תשובה: מאומה אינו קורה. אין קשר בין גדליהן. ה TLB נועד לצורך תרגום מהיר (CACHE) של כתובות וירטואליות לפיזיות, מבלי צורך לגשת לזכרון הראשי.

7. מה יקרה למספר הסיביות בשדה הכתובת הפיזית בכל כניסה של ה TLB אם גדלו של דף (בבתים) יקטן פי ארבע?
תשובה: מספר הסיביות יגדל ב שניים, משום שמספר הדפים (גם הפיזיים וגם הוירטואלים) גדל פי ארבע.

8. מה יקרה למספר הסיביות בשדה ה TAG בכל כניסה של ה TLB אם גדלו של דף (בבתים) גדל פי שניים?
תשובה: מספר הסיביות יקטן ב אחד, משום שמספר הדפים (גם הפיזיים וגם הוירטואלים) קטן פי שניים.

<u>שאלה ב'</u> (40 נק')

למעבד MIPS בן 32 סיביות שנלמד בקורס מצרפים זכרון מטמון יחיד FULLY ASSICIATIVE בעל 8 כניסות. גדלו של בלוק הינו ארבע מילים. בקר זכרון המטמון משתמש באלגוריתם LRU להחלפת בלוקים, וכן באסטרטגית WRITE-BACK. רחבו של ה MEMORY-BUS הינו 128 סיביות.
ענה **<u>במדויק ובפרוטרוט</u>** על כל אחת מהשאלות הבאות.

1. (2 נק') מה מספר סיביות ה TAG בכל כניסה של ה CACHE?
תשובה: מאחר והזכרון אסוציאטיבי מלא, כל בלוק יכול להתמקם בכל כניסת של זכרון המטמון. שתי סיביות הינן ל ארבע בתים במילה, ושתי סיביות הינן ל OFFSET של מילה ב BLOCK ולכן ל TAG נותרו $32-(2+2)=28$ סיביות.

2. (5 נק') נתון שזמן הגישה לזכרון המטמון $t_{\text{cache}}=4$ nSec, ואילו זמן הגישה לזכרון $t_{\text{mem}}=$ 50 nSec. יהי β ה HIT RATE הממוצע. א) מה צריך להיות ערכו של β על מנת שזמן הגישה הממוצע לזכרון יהיה 8 nSec? ב) מה יהיה בקירוב זמן הגישה הממוצע לזכרון אם זכרון המטמון יגדל פי שניים?
תשובה: א) $8=4+50\times(1-\beta)$, ועל כן $\beta=1-(8-4)/50=92\%$.
ב) ה MISS RATE יקטן פי שניים בקירוב ועל כן $4+50\times 0.04=6$ nSec.

3. (5 נק') נתונות הכתובות הבאות בזכרון:

0xAA08 0xA354 0xC35C 0xA358 0xAA00 0x0F1C 0xAA0C 0xAF10
0x0F14 0xFE00 0xBA08 0x5504 0x0F18
האם המלים המאוכסנות בכתובות הללו יכולות להיות מאוכסנות כולן בזכרון המטמון בו-זמנית?
תשובה: נסדר את הכתובות ברצף עולה ונחלק אותן לבלוקים איליהם הן שייכות
0x0F1C 0x0F18 0x0F14
0x5504
0xA354 0xA358
0xAA00 0xAA08 0xAA0C
0xAF10
0xBA08
0xC35C
0xFE00
מאחר ומדובר בשמונה בלוקים שונים הרי שהדבר אפשרי.

4. לפניכם קוד המבוצע עם אוגרים המאותחלים לערכים הבאים: R2=0 ,R1=0x0400, R3=0x4000. מניחים זכרון מטמון ריק בתחלת הביצוע. הפקודה הראשונה מצויה בכתובת 0x100 בזכרון הראשי.

```
loop:  L     S1, R1(R3)
       ADD   S4, S1, S2
       SUBI  R1, R1, 4
       BNE   R1, R2, loop
```

א) (18 נק') מהו ה HIT RATE במהלך בצוע הקוד הנ"ל? תאר בפירוט ומדוייק את החישוב.
תשובה: בלולאה הנ"ל ארבע פקודות שהן ארבע גישות לזכרון ועוד גישת נתונים אחת, סה"כ חמש גישות זכרון בכל לולאה.
הלולאה מבוצעת $R1/4 = 0x0400/4 = 1024/4 = 256$ פעמים.
סה"כ גישות לזכרון $5 \times 256 = 1280$.
הפקודות בלולאה יוצרות MISS בודד משום שכל ארבעת הפקודות ימצאו בבלוק אחד שיטען לזכרון המטמון פעם אחת בתחילת הלולאה וישאר שם עד לסיום התכנית בגלל השימוש באלגוריתם החלפה LRU.
הגישה לנתונים היא דרך הכתובת ב R3+R1, כאשר כל גישה הינה למילה ולכן כל ארבע גישות נדרשים לבלוק חדש, ולכן כל ארבע גישות נתונים יוצרות MISS, סה"כ $256/\mathbf{4} = \mathbf{64}$ MISSES.
לסכום, $\text{HIT RATE} = 1 - (64 + 1)/1280 = 95\%$.

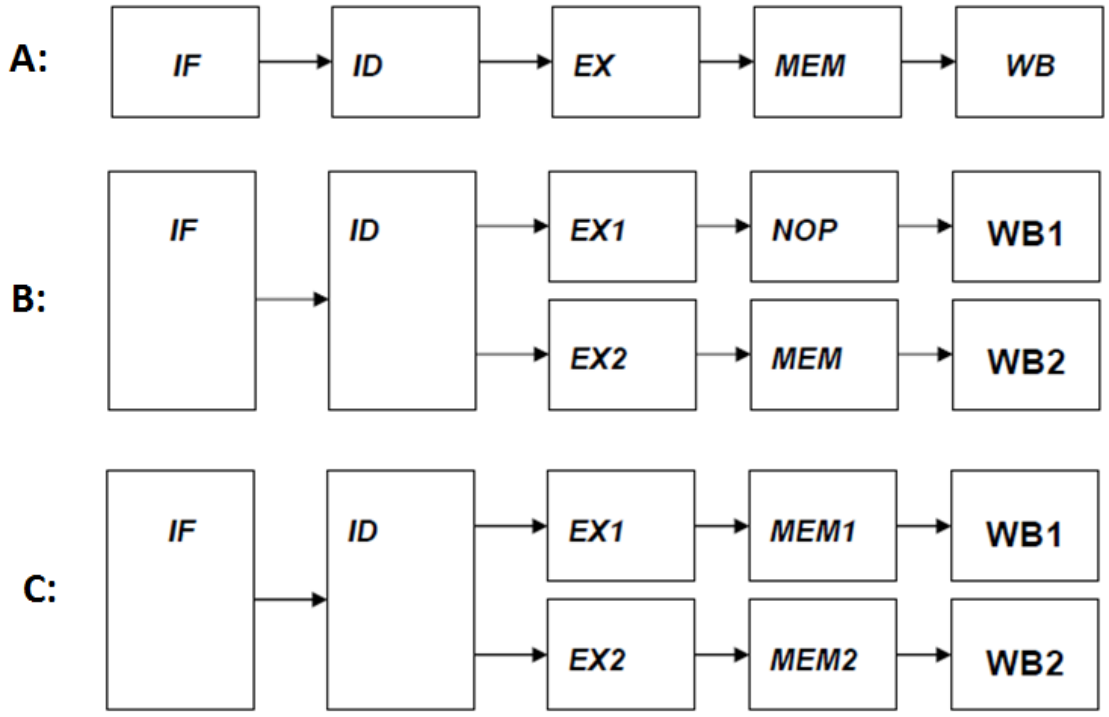
ב) (7 נק') צל"ש או טר"ש. בוגר הנדסה צעיר ונמרץ מהטכניון נדרש לממש את בקר הזכרון הנ"ל והחליט להשתמש באלגוריתם החלפה MRU (MOST RECENTLY USED) במקום LRU. חשב מחדש את ה HIT RATE.
תשובה: נוצר מצב שבו כל טעינה מחודשת של בלוק נתונים גורמת להוצאת בלוק הפקודות מזכרון המטמון. הדבר יתחיל לקרות רק אחרי שזכרון המטמון מתמלא לראשונה. מאחר ולזכרון המטמון שמונה בלוקים, והבלוק הראשון שנקרא מכיל את ארבעת הפקודות, הוצאת בלוק פקודות לראשונה מתרחשת מתרחש רק בתום $27 = 4 \times (1 - 8)$ לולאות.
סה"כ MISSES כתוצאה מנתונים איננה משתנה ושווה ל 256 כמו קודם.
סה"כ MISSES כתוצאה מפקודות $58 = \lceil (256 - 27)/4 \rceil$.
לסכום, $\text{HIT RATE} = 1 - (64 + 58 + 1)/1280 = 90\%$.

ג) (3 נק') צל"ש או טר"ש. חברו של הנ"ל, בוגר הנדסה צעיר ונמרץ מבר-אילן מציע שיטה אחרת לשיפור ביצועי הזכרון, וזאת ע"י הגדלת זכרון המטמון פי שניים וע"י כך הגדלת מספר הבלוקים פי שניים גם כן. מה ניתן לאמר כעת על ה HIT RATE ביחס לקוד הנ"ל בשימוש ב LRU?

תשובה: לא ישתנה. ברור מפתרון סעיף א' שגודל זכרון המטמון איננו משפיע כלל על ה HIT RATE.

<u>שאלה ג'</u> (40 נק')

נתונות שלש ארכיטקטורות שונות של מעבד מצונר כמתואר להלן המבצעות את קטע הקוד שבהמשך כשלצדן הפקודות רשומות כתובתן בזיכרון. (כל הפקודות בהן מופיעה האות i הינן פקודות מיידיות המשתמשות בארגומנט הערך המופיע בהן.)




```
0x40000000          lui   $t0, 0x3fff
0x40000004          ori   $t0, $t0, 0xfffc
0x40000008          lui   $t9, 0x0000
0x4000000c          ori   $t9, $t9, 0xfffc
0x40000010   loop:  lw    $t1, 0($t0)
0x40000014          lw    $t2, 0($t9)
0x40000018          lw    $t3, -4($t0)
0x4000001c          lw    $t4, -4($t9)
0x40000020          addu  $t5, $t1, $t2
0x40000024          addu  $t6, $t3, $t4
0x40000028          sw    $t5, 0($t0)
0x4000002c          sw    $t6, -4($t0)
0x40000030          addiu $t9, $t9, -8
0x40000034          bne   $t9, $0, loop
0x40000038          addiu $t0, $t0, -8
```

ארכיטקטורת **המעבדים B ו C** מאפשרות קריאה בו זמנית של שתי פקודות עוקבות ושיגור שתי פקודות עוקבות לבצוע תחת מגבלות מסוימות שלהלן.

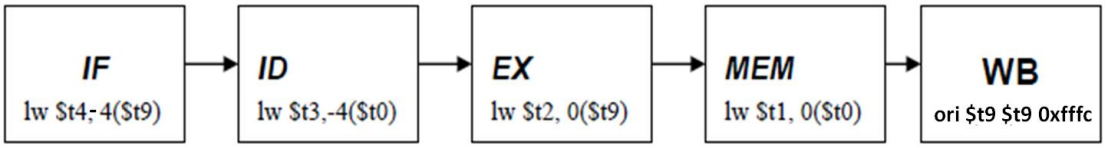
מעבד B מאפשר טפול בקפיצות לסוגיהן רק דרך הצינור 1, ופעולות זכרון רק דרך צינור 2. פעולות אריתמטיות יכולות להיות מבוצעות ע"י שני הצינורות במדה והפקודות אינן תלויות. שתי פקודות משוגרות לבצוע רק אם אינן תלויות. פקודה קודמת תבוצע ע"י צינור 1 ומאוחרת ע"י צינור 2.

למעבד C אין כל מגבלה על השמת סוגי פקודות לאחד משני הצינורות, ובכל השאר הוא דומה למעבד B.

שלשת המעבדים מזהים HAZARDS ויוצרים STALLS בשלב ה DECODE, ולשלשתם מנגנוני FORWARD. שלב ה DECODEיכול לשגר לבצוע 0, 1 או 2 פקודות, על פי התלות בין הפקודות ובמשאבי החומרה. לכל המעבדים מנגנון חיזוי קפיצות עם טבלת הסטוריה בת שתי סיביות. הפעם הראשונה שמתרחשת קפיצה היא נחזית כ TAKEN.

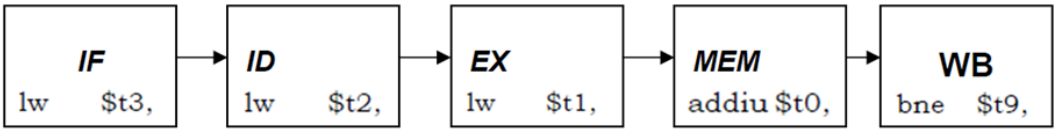
במדה ופקודה לא שוגרה לבצוע, NOP יתפוס את מקומה והיא תבוצע בהמשך.

במעבד A, כאשר הפקודה שבכתובת 0x4000000**c** נמצאת בשלב ה WB, הצינור נראה כדלהלן.

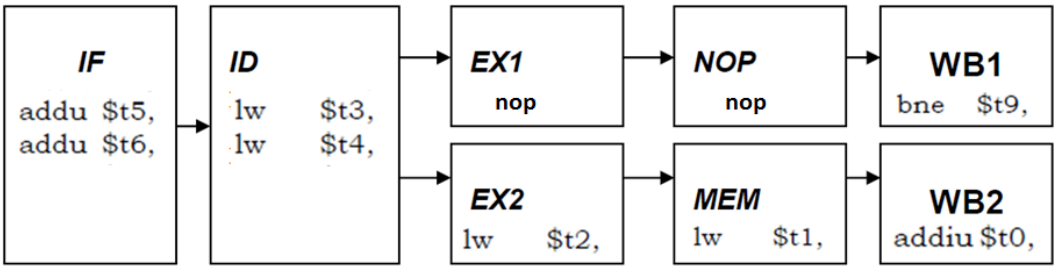


1. (20 נק') נניח כעת הפקודה המצויה בכתובת 0x4000003**4** מצויה כבר במהלך הלולאה השנייה בשלב ה WB (מעבד A) או ב WB1 (מעבדים B ו C). רשום על גבי טופס הבחינה בתוך כל שלב בצינור של כל אחד מהמעבדים את הפקודה המצויה בתוכו בזמן הנ"ל (מספיק לרשום רק את שני השדות הראשונים של הפקודה).

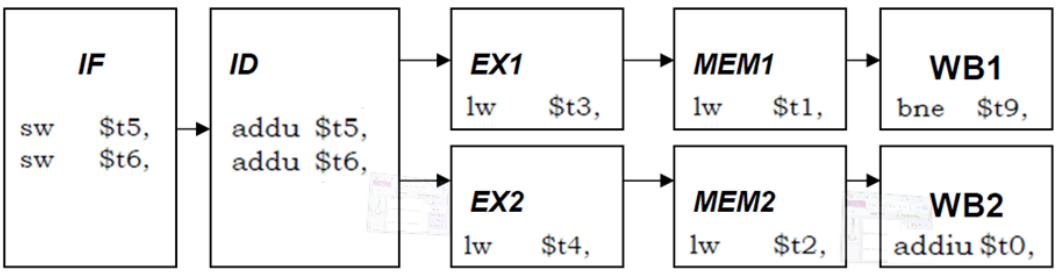
Processor A:



Processor B:



Processor C:



טעות נפוצה: התעלמות מהפקודה addiu $t0, $to, -8 הרשומה בסוף הלולאה. זוהי כמובן חלק ממנה ונכנסת לצנור בטרם פקודת הקפיצה המותנית מקבלת החלטה.

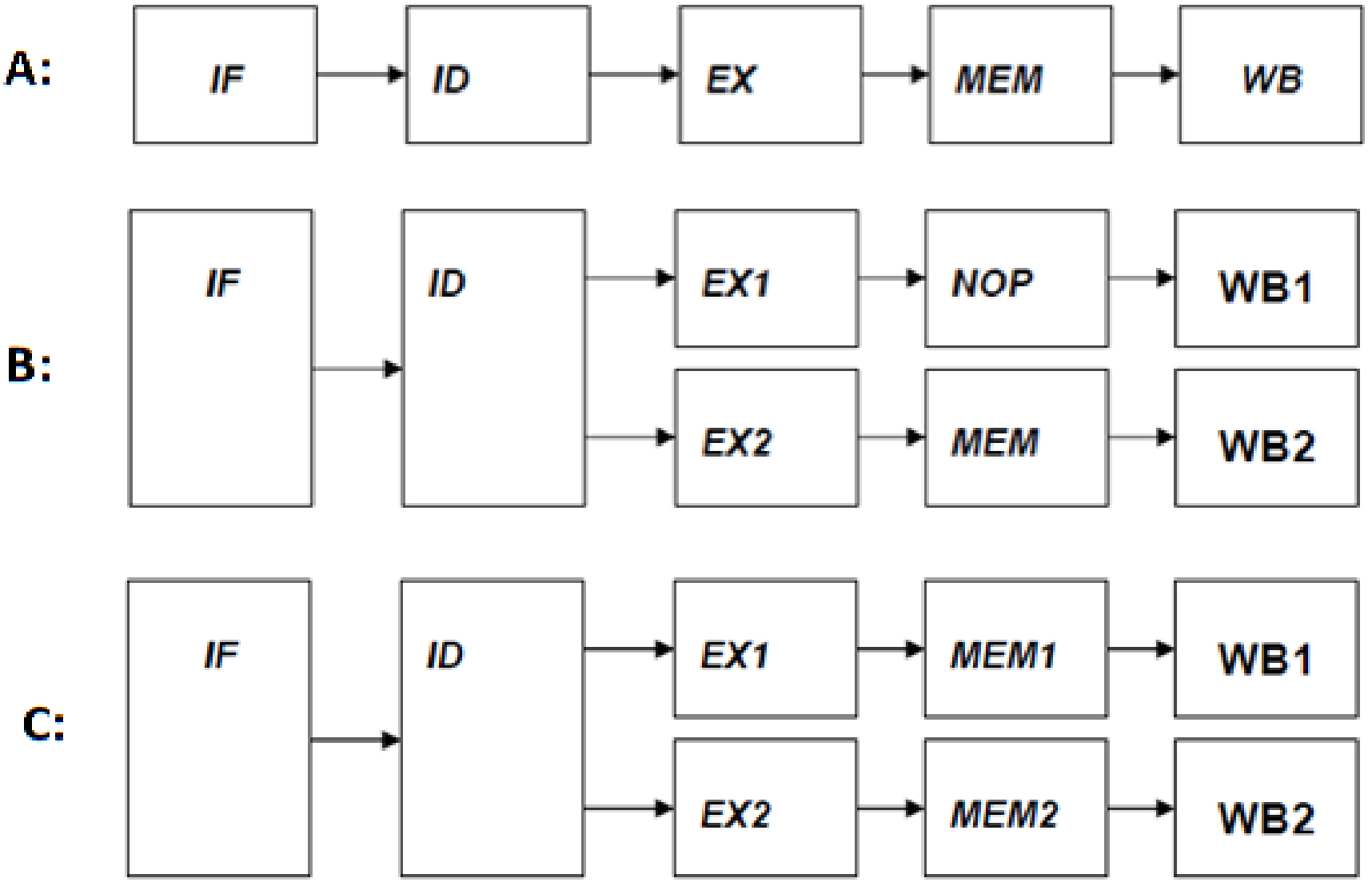
2. (4 נק') נדרש לסדר מחדש את הפקודות שבקוד על מנת לשפר את ביצועי התכנית. לאיזו מבין הארכיטקטורות B או C בדבר קל יותר לבצוע, ומדוע?

תשובה: ארכיטקטורה C הינה הנוחה ביותר משום שמבין השלש היא בעלת פחות מגבלות שיש לקחת בחשבון בעת אופטימיזציה של סדר בפקודות.

3. (16 נק') סדרו מחדש את הפקודות באופן אופטימלי לכל אחד משני המעבדים בתוך הטבלה המצורפת, וזאת מבלי לשנות את גודל הקוד. במידה וישנם STALLS רשמו על יד הטבלה, משמאל לפקודה המתאימה, היכן הם קורים.

	Address	Label	Instruction
	0x40000000		Lui $t0, 0x3fff #optimization for B
	0x40000004		lui $t9, 0x0000
Stall	0x40000008		ori $t0, $t0, 0xfffc
Stall	0x4000000c		ori $t9, $t9, 0xfffc
	0x40000010	Loop:	addiu $t9, $t9, -8
	0x40000014		lw $t1, 0($t0)
	0x40000018		addiu $t0, $t0, -8
	0x4000001c		lw $t2, 8($t9)
	0x40000020		lw $t3, 4($t0)
NOP occurs	0x40000024		lw $t4, 4($t9)
	0x40000028		addu $t5, $t1, $t2
	0x4000002c		sw $t5, 0($t0)
	0x40000030		addu $t6, $t3, $t4
	0x40000034		bne $t9, $0, loop
	0x40000038		sw $t6, -4($t0)

	Address	Label	Instruction
	0x40000000		lui $t0, 0x3fff #optimization for C
	0x40000004		lui $t9, 0x0000
Stall	0x40000008		ori $t0, $t0, 0xfffc
Stall	0x4000000c		ori $t9, $t9, 0xfffc
	0x40000010	loop:	lw $t3, -4($t0)
	0x40000014		lw $t4, -4($t9)
	0x40000018		lw $t1, 0($t0)
	0x4000001c		lw $t2, 0($t9)
	0x40000020		addiu $t9, $t9, -8
	0x40000024		addiu $t0, $t0, -8
	0x40000028		addu $t6, $t3, $t4
	0x4000002c		addu $t5, $t1, $t2
	0x40000030		sw $t5, 8($t0)
	0x40000034		bne $t9, $0, loop
	0x40000038		sw $t6, 4($t0)


A:
IF
ID
EX
MEM
WB
B:
IF
ID
EX1
NOP
WB1
EX2
MEM
WB2
C:
IF
ID
EX1
MEM1
WB1
EX2
MEM2
WB2

Address	Label	Instruction C processor
0x40000000		
0x40000004		
0x40000008		
0x4000000c		
0x40000010		
0x40000014		
0x40000018		
0x4000001c		
0x40000020		
0x40000024		
0x40000028		
0x4000002c		
0x40000030		
0x40000034		
0x40000038		
0x4000003C		
0x40000040		
0x40000044		
0x40000048		

Address	Label	Instruction C processor
0x40000000		
0x40000004		
0x40000008		
0x4000000c		
0x40000010		
0x40000014		
0x40000018		
0x4000001c		
0x40000020		
0x40000024		
0x40000028		
0x4000002c		
0x40000030		
0x40000034		
0x40000038		
0x4000003C		
0x40000040		
0x40000044		
0x40000048		