אוניברסיטת בר-אילן (ע"ר)
הפקולטה להנדסה
רמת-גן 52900

BAR-ILAN UNIVERSITY (RA)
Faculty of Engineering
Ramat-Gan 52900, Israel

Tel: 03-5317722

engbi@mail.biu.ac.il

תכן לוגי

תשע"ח סמסטר קיץ מועד ב'

83-253

מרצה: פרופ' שמואל וימר

מתרגלים: מר בנימין פרנקל ומר גילעד פלן

- **יש לקרוא היטב את ההוראות.**
- **חובה לענות על כל השאלות.**
- **סך הנקודות בבחינה הוא 110 אך הציון הסופי בבחינה לא יעלה על 100.**
- **יש לנמק את כל תשובותיכם.**
- יש לשרטט באופן ברור, להוסיף הסבר מילולי, ובמידת הצורך לצייר את החלק הרלוונטי במחברת הבחינה.
- יש להקפיד על כתב יד קריא!
- **חומר עזר מותר בשימוש**: מחשבון, ספר הקורס ושקפים מההרצאות בלבד (כולל הערות על גביהם).
- **משך הבחינה:** שלוש שעות.
- **יש לצרף את שאלוני הבחינה למחברת!**

בהצלחה!

<u>שאלה מספר 1</u> (25 נקודות):

בשאלה זו נעסוק בארכיטקטורת Multi-Cycle MIPS.

כפי שלמדנו במהלך הקורס, הפקודה addi סוכמת את הערך המופיע בשדה ה- immediate עם הערך השמור ברגיסטר $rs ושומרת את הסכום ברגיסטר $rt, דהיינו:

addi $t2, 15($t1) # $t2 ← <$t1> + 15

א. **בתרשים מספר 1** מתוארת מכונת מצבים עבור Multi-Cycle MIPS. הוסף על גבי התרשים את כל הדרוש על מנת לתמוך גם בפקודה addi. יש להוסיף מספר מינימלי של מצבים!

- כעת, רוצים לשפר את ביצועי המעבד ע"י ביצוע Pre-Fetch, כך שבפקודות אשר אינן עושות שימוש בזיכרון במחזור השעון האחרון שלהן תתבצע פעולת ה- Fetch של הפקודה הבאה.

ב. באילו סוגי פקודות לא ניתן לבצע Pre-Fetch? הסבר בפירוט מדוע.

פקודת SW משתמשת בחזור שעון האחרון שלה בזיכרון, ולכן לא ניתן לעשות Pre-Fetch, משום שלא ניתן לגשת לזיכרון לצורך קריאה.

פקודת Jump וכן פקודת Branch כותבות ל- PC במחזור שעון האחרון שלהן, ולכן לא ניתן לבצע Pre-Fetch, משום שאי-אפשר לעשות PC=PC+4 כמו ב- Fetch רגיל.

בנוסף, פקודת Branch משתמש ב- ALU במחזור שעון האחרון שלה, ולכן לא ניתן להשתמש בו לצורך חישוב של PC+4.

ג. בהנחה שחלוקת הפקודות היא לפי הטבלה הבאה, מה יהיה שיפור הביצועים (Speedup) של המעבד (בהנחה שהוא ימשיך לעבוד באותו תדר שעון)?

סוג פקודה	אחוזים מסך הפקודות
Load	25%
Store	10%
Branch	11%
Jump	2%
(math) R-Type	27%
(math) I-Type	25%

שימו לב, כי יש להחסיר מחזור שעון אחד מכל פקודה שכן יכולה לעשות Pre-Fetch. אמנם, מחזור השעון נחסך מהפקודה הבאה, אבל זה לא משנה את החישוב.

$$CPI_{old} = 0.25 \times 5 + 0.1 \times 4 + 0.11 \times 3 + 0.02 \times 3 + 0.27 \times 4 + 0.25 \times 4 = 4.12$$

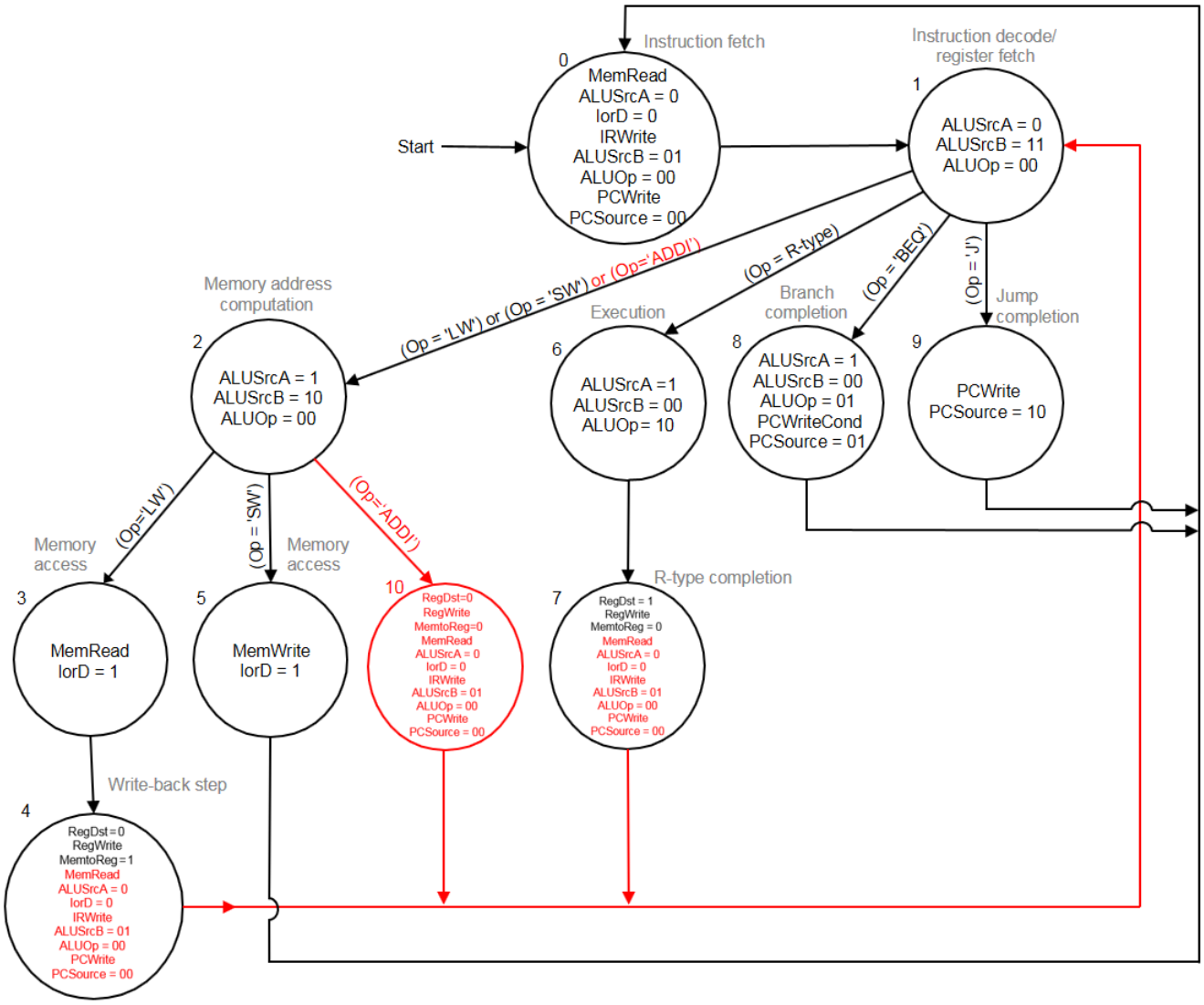
$$CPI_{new} = 0.25 \times 4 + 0.1 \times 4 + 0.11 \times 3 + 0.02 \times 3 + 0.27 \times 3 + 0.25 \times 3 = 3.35$$

$$Speedup = \frac{4.12}{3.35} \cong 1.23$$

ד. האם נדרשים שינויים ב- Data Path על מנת לבצע שינוי זה? הסבר בפירוט!

לא, מכיוון שמצב Fetch ממילא ממומש, ומכיוון שהרכיבים בהם אנחנו משתמשים (הזיכרון וה-PC) פנויים באותו מחזור שעון מלכתחילה, לכן אין צורך בשינויים ב- Data Pass. אולם, יש צורך בשינויים בקווי הבקרה.

ה. רשום את כל השינויים הנדרשים במכונת המצבים על מנת לתמוך בשינוי זה (Pre-Fetching). שים-לב, ניתן לשנות מצבים וקווים קיימים. את תשובתך לסעיף זה יש לרשום במחברת הבחינה, ולא על-גבי תרשים מספר 1.



<u>שאלה מספר 2</u> <u>(55 נקודות):</u>

בשאלה זו נעסוק במערכת סינכרונית המחשבת את האיבר ה- N בסדרת פיבונאצ'י.

סדרת פיבונאצ'י (Fibonacci) היא הסדרה ששני איבריה ההתחלתיים הם 1, 0 וכל איבר לאחר מכן שווה לסכום שני קודמיו. בהתאם לכך, איבריה הראשונים של הסדרה הם:

0, 1, 1, 2 ,3, 5, 8, 13, 21, 34, 55, 89, 144, …

ה- Data Path של המערכת מתואר **בתרשים מספר 2**. המערכת מקבלת In אשר מגדיר את האיבר המבוקש בסדרה (לא כולל שני האיברים הראשונים המשמשים לחישוב ההתחלתי). המערכת מחשבת את האיבר המבוקש ומדליקה אות בקרה 'Done' כאשר האיבר הרצוי מופיע ב- Out.

לדוגמא: עבור In=1 המערכת תחשב Out=1, עבור In=2 המערכת תחשב Out=2, עבור In=3 המערכת תחשב Out=3, עבור In=4 המערכת תחשב Out=5, וכו'.

לאחר שהמערכת הדליקה את אות ה- 'Done' היא מוכנה לקלוט מספר חדש (גדול מאפס) ולבצע חישוב מחדש. כל עוד יתקבל In=0 המערכת תמתין במצב ההתחלתי, ובמוצא יופיע ה- Out הקודם.

קו הבקרה ALUOp הינו כניסה לרכיב ה- ALU, כאשר עבור $ALUOp = 0$ יתבצע החיבור $Y + X$, ועבור $ALUOp = 1$ יתבצע החיסור $Y - X$.

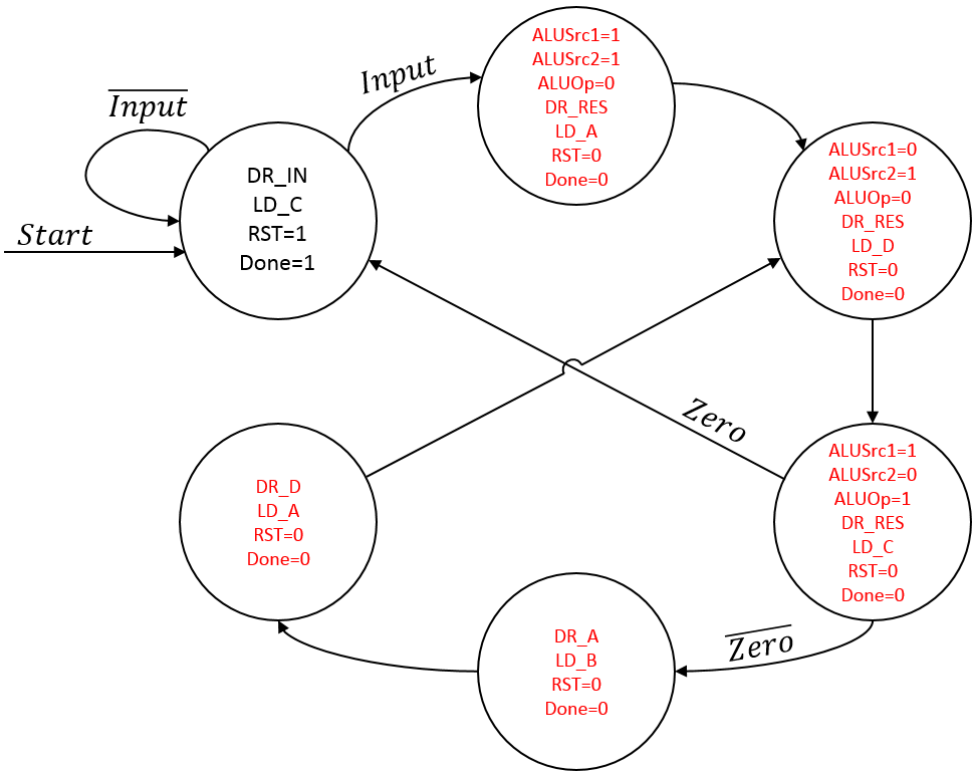
קו החיווי Zero הינו מוצא של ה- ALU אשר מקבל ערך '1' במקרה שבו תוצאת החישוב הינה אפס.

שימו-לב כי האות RST מאפס אך ורק את רגיסטר B.

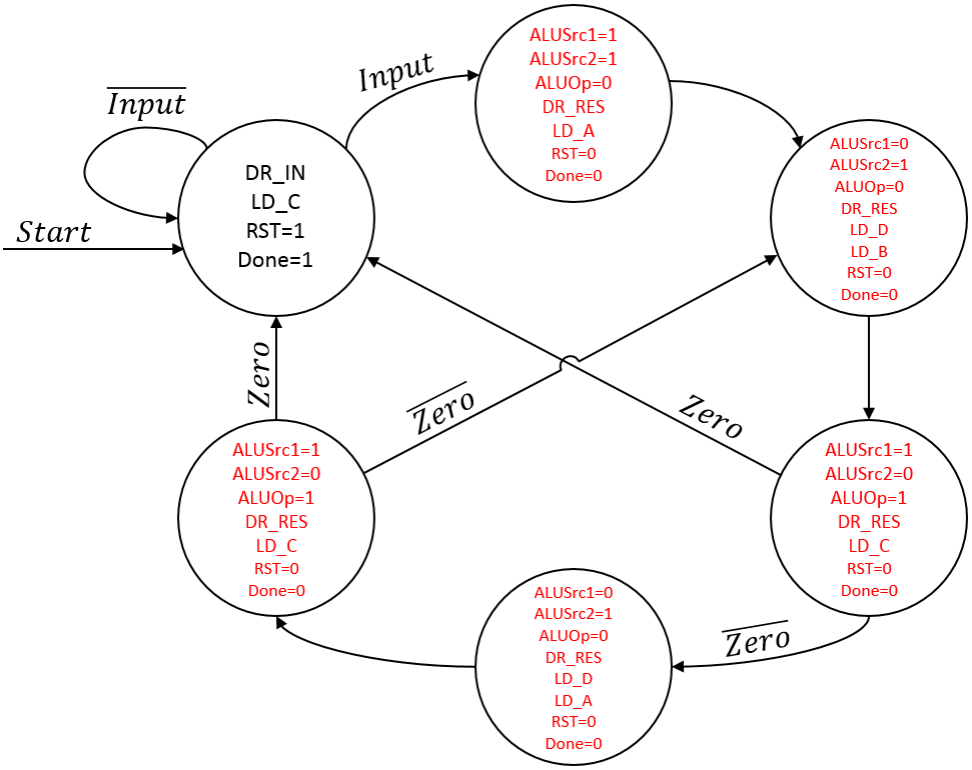
א. עליכם להשלים את דיאגרמת המצבים הבאה עבור המערכת (אין להוסיף מצבים).

להלן נציג שני פתרונות, כאשר לכל פתרון ישנם יתרונות וחסרונות כפי שניתן לראות בהמשך:

פתרון ראשון:



פתרון שני:



ב. נניח שבמחזור שעון מספר 1 מתקבל ערך $In = N \neq 0$ בכניסה למערכת. כמה מחזורי שעון ייקח למערכת לסיים את החישוב?

עבור הפתרון הראשון: יחלפו $\boldsymbol{4N}$ מחזורי שעון עד שסיגנל ה- Done יעיד על סיום החישוב (שימו-לב שבמחזור השעון שבו done עולה ל- '1' המערכת כבר דרוכה ומוכנה לקבלת כניסה נוספת).

עבור הפתרון השני: יחלפו $\boldsymbol{2N + 2}$ מחזורי שעון עד שסיגנל ה- Done יעיד על סיום החישוב.

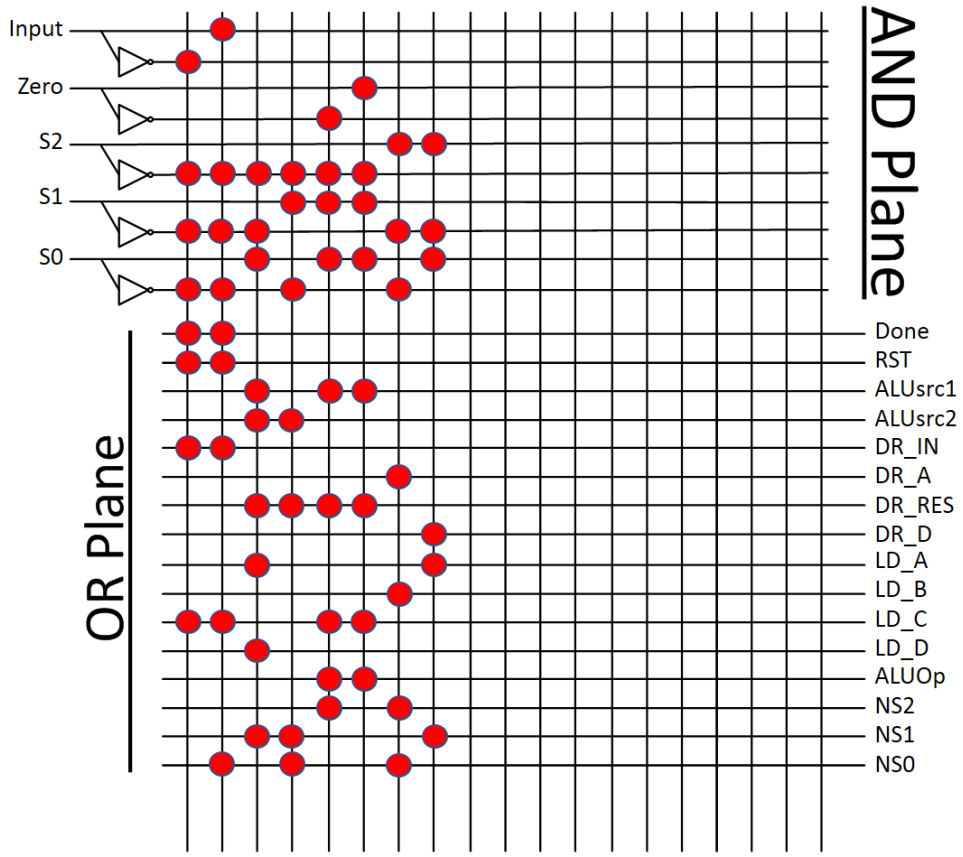
ג. בהנחה שהמערכת עובדת עם אוגרים של 8 סיביות, מהו ה- N המקסימלי שהמערכת יכולה לקלוט ולבצע את החישוב? האם תשובתך תלויה ב- Signed / Unsigned? פרט.

אם מדובר במערכת המייצגת את המספרים בייצוג Unsigned הרי שהמספר המקסימלי שהמערכת יכולה לייצג הוא 255, ולכן, ה- N המקסימלי שהמערכת תוכל לקלוט ולבצע את החישוב הוא $N = 12$, משום שתוצאת החישוב תהיה 233. את האיבר הבא בסדרת הפיבונאצ'י (377) המערכת כבר לא תוכל לייצג בעזרת 8 סיביות.

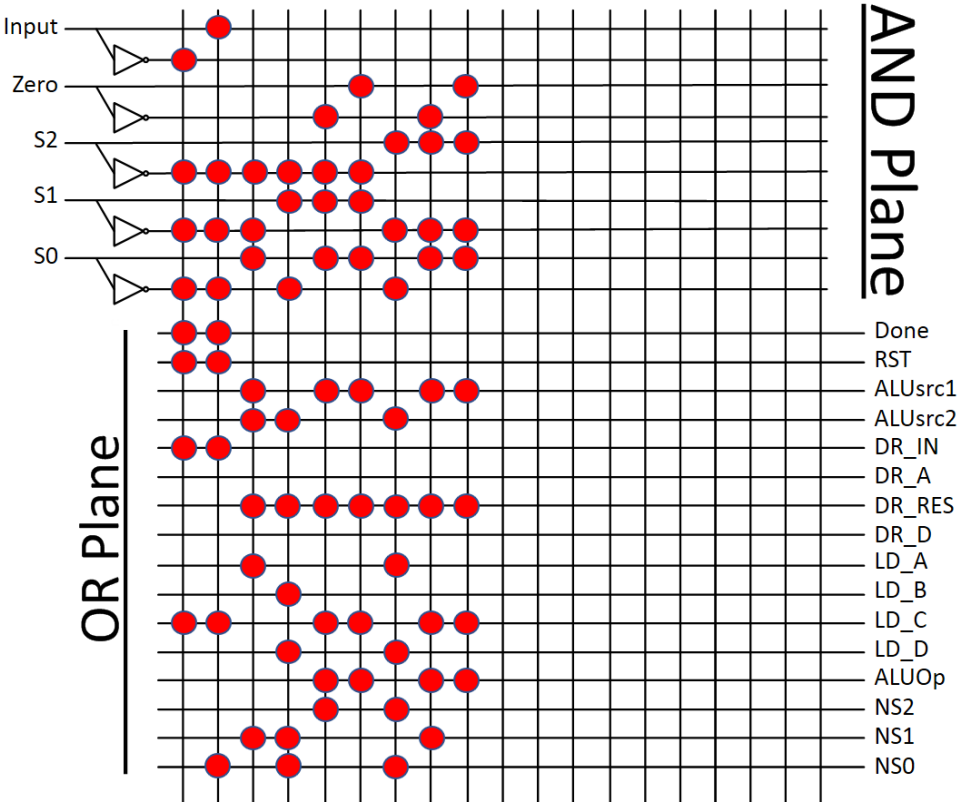
אם מדובר במערכת המייצגת את המספרים בייצוג Signed הרי שהמספר המקסימלי שהמערכת יכולה לייצג הוא 127, ולכן, ה- N המקסימלי שהמערכת תוכל לקלוט ולבצע את החישוב הוא $N = 10$, משום שתוצאת החישוב תהיה 89. את האיבר הבא בסדרת הפיבונאצ'י (144) המערכת כבר לא תוכל לייצג בעזרת 8 סיביות.

ד. ממשו את הבקר (Controller) של המערכת בעזרת PLA:

עבור הפתרון הראשון:



עבור הפתרון השני:



ה. מהו גודלו של ה- PLA (במונחי Gates)?

גודל PLA מאופיין ע"י שלושה גורמים: מספר הכניסות, מספר המכפלות ומספר היציאות.

במקרה שלנו, עבור הפתרון הראשון: ישנן 5 כניסות, 8 מכפלות ו- 16 יציאות.

$$(2 \times 5 + 16) \times 8 = 208\ Gates$$

עבור הפתרון השני: ישנן 5 כניסות, 9 מכפלות ו- 14 יציאות (אין צורך ב- DR_A, DR_D).

$$(2 \times 5 + 14) \times 9 = 216\ Gates$$

התקבל גם החישוב: (כניסות + יציאות) × (מכפלות), היות ומה שחשוב זו ההבנה של הגורמים המשפיעים על השטח, ולא איזה חישוב מדויק, שממילא תלוי בהמון פרמטרים פיזיים/טכנולוגיים.

ו. כתוב תכנית בשפת אסמבלי המממשת את פעולת המערכת הנ"ל, כאשר הערך $In = N$ (האיבר בסדרת הפיבונאצ'י שאותו יש לחשב) שמור בזיכרון הנתונים, כאשר כתובתו של הערך הזה שמורה ברגיסטר $t1. את התוצאה הסופית של החישוב יש לשמור ב- $t7 (רק את התוצאה הסופית, ולא את חישובי הביניים). רשום את הקוד בתוספת הסבר עבור כל שורה ושורה.

לדוגמא, יש לרשום את השורה הראשונה כך:

	$lw\ \$t2, 0(\$t1)$	$t2 \leftarrow mem[t1]$	$t2\ is\ the\ Counter\ register\ (C)$
	$addi\ \$t4, \$zero, 0$	$t4 \leftarrow 0$	$reset\ for\ t4\ (B)$
	$addi\ \$t3, \$zero, 1$	$t3 \leftarrow 1$	$set\ for\ t3\ (A)$
$Loop$:	$add\ \$t5, \$t3, \$t4$	$t5 \leftarrow t3 + t4$	$D \leftarrow A + B$
	$addi\ \$t2, \$t2, (-1)$	$t2 \leftarrow t2 - 1$	$C \leftarrow C - 1$
	$add\ \$t4, \$t3, \$zero$	$t4 \leftarrow t3$	$B \leftarrow A$
	$add\ \$t3, \$t5, \$zero$	$t3 \leftarrow t5$	$A \leftarrow D$
	$bne\ \$t2, \$zero, Loop$	$t3 \leftarrow t5$	$if\ C \neq 0\ then\ go\ back\ to\ Loop$
$Done$:	$add\ \$t7, \$t5, \$zero$	$t7 \leftarrow t5$	$Out\ Register \leftarrow D$

ז. כמה מחזורי שעון ייקח לתכנית שכתבת בסעיף הקודם לרוץ על מעבד Single-Cycle MIPS וכמה מחזורים ייקח לה לרוץ על Multi-Cycle MIPS ? פרט.

ב- $Single\ Cycle\ MIPS$ כל פקודה לוקחת מחזור שעון אחד, ולכן, התכנית תארך:

$$\mathbf{4 + 5 \times N\ [cycles]}$$

לעומת זאת, ב- *Multi Cycle MIPS*:

Instruction	#Cycles
lw	5
add	4
addi	4
bneq	3

ולכן, התכנית תארך:

$$\Rightarrow 5 + 4 + 4 + N \cdot (4 + 4 + 4 + 4 + 3) + 4 = \mathbf{17 + 19N\ [cycles]}$$

אם נפעיל את מנגנון ה- Pre-Fetch כפי שלמדנו בשאלה מספר 1 הרי שנקבל:

Instruction	#Cycles
lw	4
add	3
addi	3
bneq	3

ולכן, התכנית תארך:

$$\Rightarrow 4 + 3 + 3 + N \cdot (3 + 3 + 3 + 3 + 3) + 3 = \mathbf{13 + 15N\ [cycles]}$$

<u>שאלה מספר 3</u> (30 נקודות):

בשאלה זו נעסוק בזיכרונות מטמון (cache).

נתונים שלושה זיכרונות מטמון (caches), אשר עובדים בשיטת direct mapped, כל אחד מהם מפרש את הכתובת בעלת 8 הסיביות באופן קצת שונה, בהתאם לפורמט {tag : setIdx : byteOffset} אשר מוגדר לכל אחד מהם.

א. מלא את הטבלה הבאה בהתאם לפורמט המוגדר עבור כל cache:

Address Formats →	C1 {2:2:4}	C2 {2:3:3}	C3 {2:4:2}
Block Size (Bytes):	16	8	4
Cache Size (Blocks):	4	8	16

ב. עבור כל אחת מהכתובות ברצף הגישות הבא, עליך לציין האם היה hit (H) או שהיה miss (M):

Address References (in binary)			
00000010	M	M	M
00000100	H	H	M
00001000	H	M	M
00010000	M	M	M
00100000	M	M	M
01000000	M	M	M

- כעת, נתון זיכרון בעל 3 רמות היררכיה (tree-level hierarchy), המכיל את הרמות הבאות: L1 המגובה ע"י L2 המגובה ע"י ה- Main Memory.

ג. עבור כל אחת מהכתובות ברצף הגישות הבא עליך לציין האם היה hit (H) או שהיה miss (M) בכל אחד מזיכרונות המטמון (caches). אם בכלל לא הייתה גישה לזיכרון המטמון – ציין זאת במשבצת המתאימה בעזרת קו נטוי. בנוסף, עבור כל hit או miss, עליך לציין באיזה set המאורע התרחש, לדוגמא: יש לסמן M→A עבור miss שהתרחש ב- set A.

	L1	L2
Sets	16	16
Bytes/Block	16	256
Ways	2	2
0xABCDE	*M → D*	*M → C*
0xABCD0	*H → D*	–
0xABDD0	*M → D*	*M → D*
0xABCE0	*M → E*	*H → C*
0xABCEF	*H → E*	–
0xABCDE	*H → D*	–

ד. מהו ה- miss rate של L1 ושל L2 ?

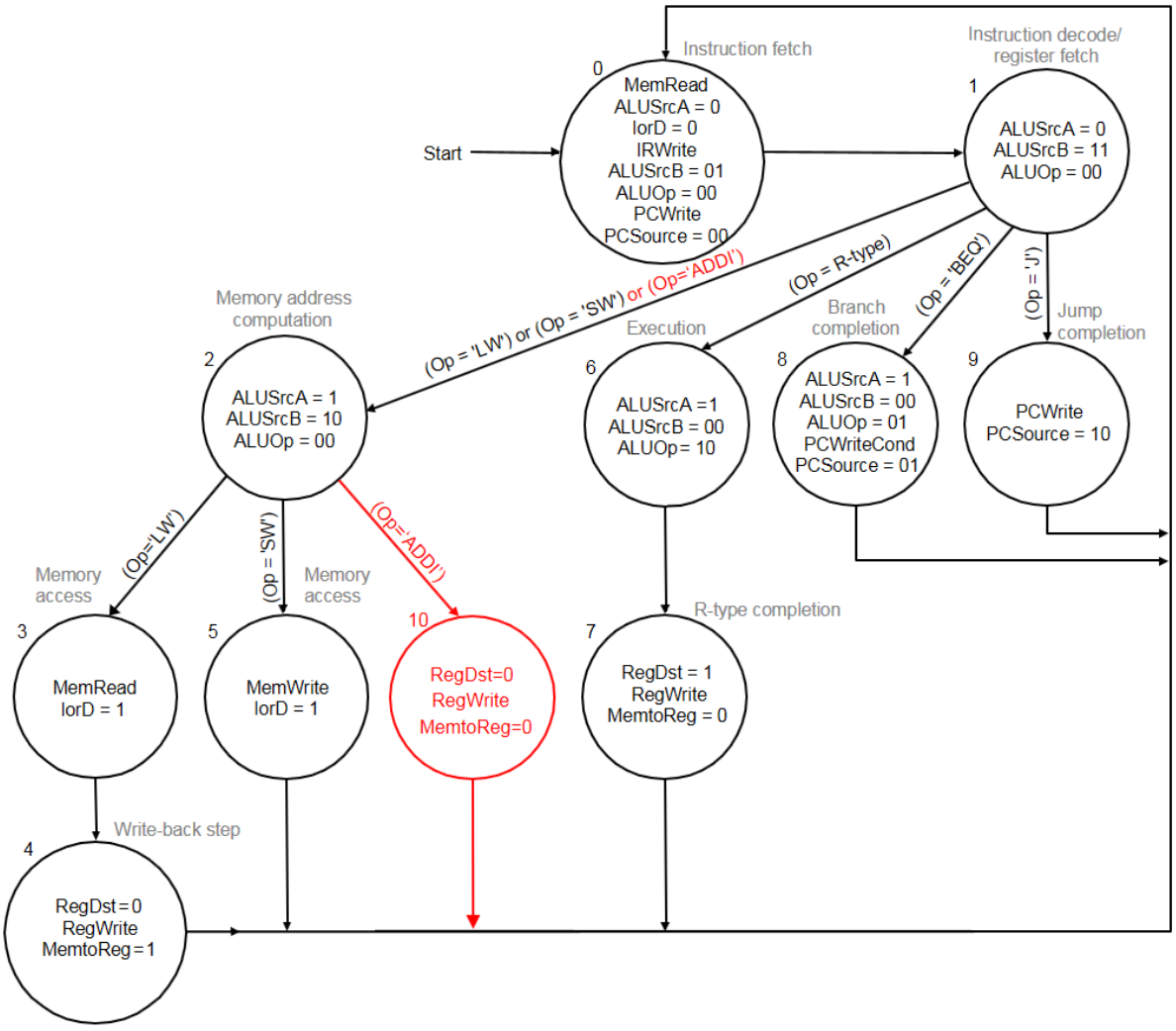
$$MR_{L1} = 50\% \, , MR_{L2} = 66.6\%$$

- נתון כי זמן הגישה ל- L1 הוא 5 מחזורי שעון, זמן הגישה ל- L2 הוא 50 מחזורי שעון, וזמן הגישה ל- Main Memory הוא 1000 מחזורי שעון.

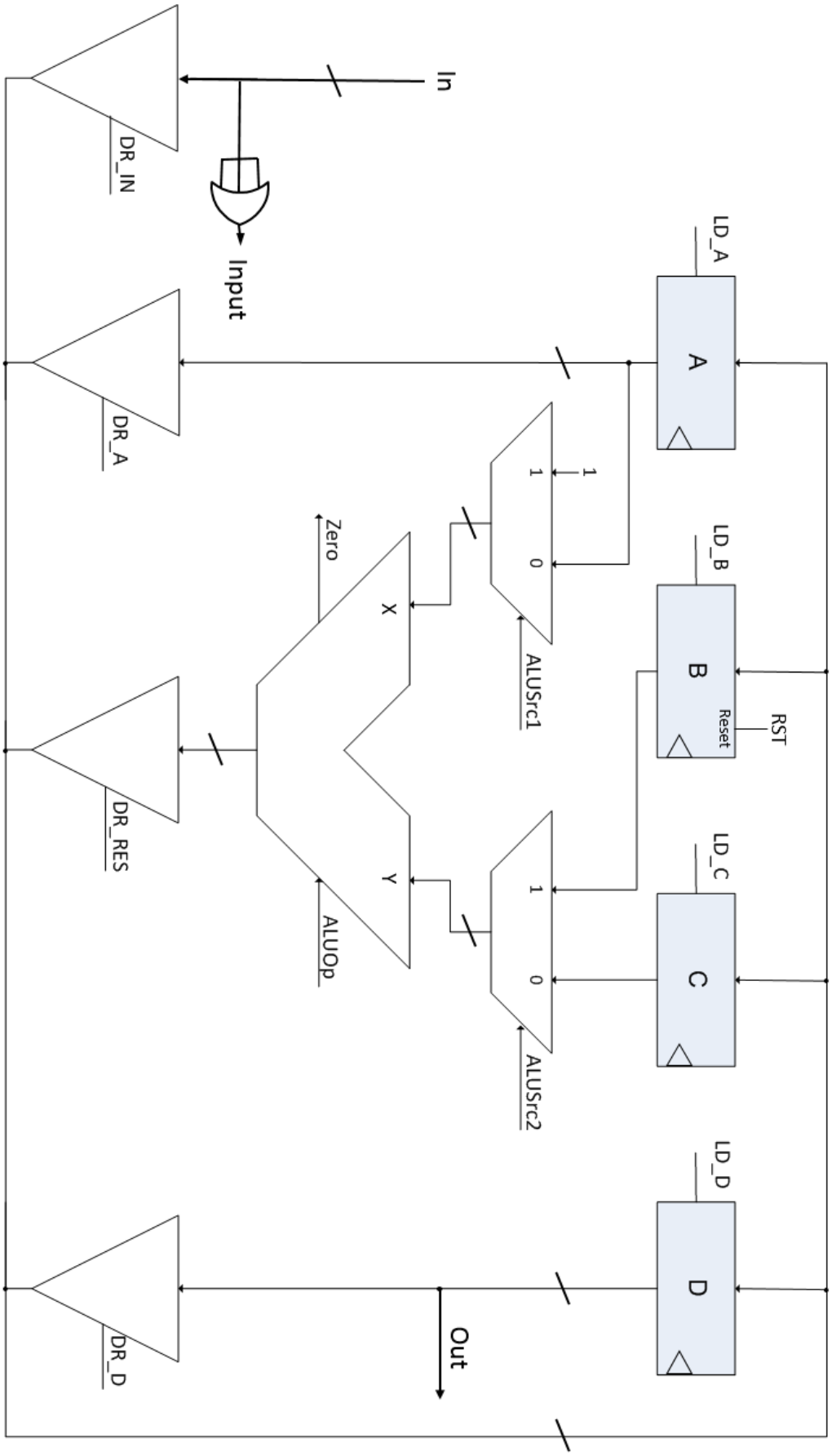
ה. מהו זמן הגישה הממוצע (במחזורי שעון) עבור הזיכרון כולו (AMAT) בהנחה שרצף הגישות לזיכרון הוא כמו בסעיף ג?

$$\text{AMAT} = T_{L1} + MR_{L1} \times (T_{L2} + MR_{L2} \times T_{Mem}) =$$
$$= 5 + 0.5 \times (50 + 0.666 \times 1000) = 363$$

תרשים מספר 1


Start
0
MemRead
ALUSrcA = 0
IorD = 0
IRWrite
ALUSrcB = 01
ALUOp = 00
PCWrite
PCSource = 00
Instruction fetch
1
ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00
Instruction decode/
register fetch
(Op = 'LW') or (Op = 'SW') or (Op='ADDI')
(Op = R-type)
(Op = 'BEQ')
(Op = 'J')
Memory address
computation
2
ALUSrcA = 1
ALUSrcB = 10
ALUOp = 00
Execution
6
ALUSrcA =1
ALUSrcB = 00
ALUOp= 10
Branch
completion
8
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCWriteCond
PCSource = 01
Jump
completion
9
PCWrite
PCSource = 10
(Op='LW')
(Op = 'SW')
(Op='ADDI')
Memory
access
Memory
access
3
MemRead
IorD = 1
5
MemWrite
IorD = 1
10
RegDst=0
RegWrite
MemtoReg=0
R-type completion
7
RegDst = 1
RegWrite
MemtoReg = 0
Write-back step
4
RegDst = 0
RegWrite
MemtoReg = 1

תרשים מספר 2


In
Input
DR_IN
DR_A
DR_RES
DR_D
LD_A
LD_B
LD_C
LD_D
A
B
C
D
RST
Reset
ALUSrc1
ALUSrc2
ALUOp
Zero
X
Y
Out