סיכום ונוסחאון בארגון המחשב

קורס 20471 · עודכן ב-2026ג׳

(נכתב על ידי דניאל וונג – www.wongi.io)

כל סוגי השאלות במבחנים של הקורס, לפי סדר המבחן: מה צריך לדעת (עם העמוד במדריך הלמידה), תבנית פתרון בצעדים, טבלת עבודה פתורה, המלכודות שעלו נקודות ובאילו מבחנים זה נשאל. כל מספר בדוגמאות חושב בכלי: MARS, טבלאות אמת ושברים מדויקים.

תוכן העניינים

נושא	עמוד	נושא	עמוד
ביצועים	1	המעבד החד-מחזורי	15
ייצוג מספרים	3	הצנרת	19
לוגיקה ורכיבים	5	זיכרון מטמון	25
שפת אסמבלי MIPS	10		

ביבליוגרפיה

בוסני, ר. (2023). *ארגון המחשב – מדריך למידה* (20471). רעננה: האוניברסיטה הפתוחה. ״מדריך עמ׳ N״ בסיכום הוא העמוד המודפס במדריך.

Patterson, D. A., & Hennessy, J. L. (2021). *Computer Organization and Design: The Hardware/Software Interface, MIPS Edition* (6th ed.). Morgan Kaufmann.

1 ביצועים

1.1 ההגדרות והנוסחאות

ניסוח	כתיב
ביצועים וזמן ריצה – ביצועים $\frac{1}{\text{זמן ריצה}} =:$. **מדד ההאצה** (speedup) הוא זמן האיטי חלקי זמן המהיר, ולכן תמיד גדול מ-1. (מדריך עמ' 11)	$Speedup = \frac{T_{slow}}{T_{fast}} > 1$
שעון – CCT זמן מחזור שעון [sec/cycle], CR קצב (תדר) השעון [Hz = cycle/sec], והם הפוכים זה לזה. לזמן ולתדר – קידומות עשרוניות; לגודל זיכרון – קידומות בינריות. (עמ' 11–12)	$2\,GHz \to 0.5\,ns : CCT = \frac{1}{CR}$ $G = 10^9,\ n = 10^{-9},\ p = 10^{-12}$
זמן מעבד – מספר מחזורי השעון של התוכנית כפול זמן מחזור. IC – מספר הפקודות שבוצעו בפועל [ins/program], CPI – מחזורים לפקודה בממוצע [cycle/ins]. (עמ' 13–14)	$CC = IC \cdot CPI$ $\text{CPU time} = IC \cdot CPI \cdot CCT = \frac{IC \cdot CPI}{CR}$
CPI ממוצע – סכום משוקלל של ה-CPI של קבוצות הפקודות, w_i החלק היחסי של הקבוצה ($\sum w_i = 1$). באותה ארכיטקטורה (ISA) ואותה תוכנית מכונה – אותו IC. (עמ' 14)	$CPI = \sum_{i=1}^{n} CPI_i \cdot w_i$
MIPS – מיליוני פקודות בשנייה. מדד מטעה: לא מתחשב ב-ISA (מכונה עם פקודות פשוטות מבצעת יותר פקודות) ומשתנה מתוכנית לתוכנית. (עמ' 19)	$MIPS = \frac{IC}{T \cdot 10^6} = \frac{CR}{CPI \cdot 10^6}$
חוק אמדל – F החלק **מזמן הריצה** שמשופר, S פי כמה הוא משופר. גם אם $S \to \infty$ ההאצה חסומה: "make the common case fast". (עמ' 18)	$T_{new} = \frac{T_{affected}}{S} + T_{unaffected}$ $Speedup = \frac{1}{(1-F)+\frac{F}{S}} \le \frac{1}{1-F}$

1.2 זמן ריצה, CPI ממוצע והשוואת מכונות

תבנית – "מה זמן הריצה / ה-CPI / איזו מכונה מהירה ובכמה"

1. **רשימת נתונים** עם יחידות: IC, CR או CCT, החלקים w_i (או מספר הפקודות בכל קבוצה) ו-CPI_i. תדר נתון – הופכים ל-CCT.
2. **טבלת הקבוצות** – עמודה $w_i \cdot CPI_i$, והסכום בשורה האחרונה הוא ה-CPI הממוצע. נתון מספר פקודות לקבוצה במקום אחוז – סופרים מחזורים $\sum IC_i \cdot CPI_i$ ומחלקים ב-IC.
3. **זמן** – $T = IC \cdot CPI / CR$, ושורה אחת עם היחידות בסוגריים כמו במדריך.
4. **השוואה** – מחשבים T לכל מכונה ומחלקים: איטי חלקי מהיר. לא משווים CPI או תדר לבד.
5. **"מה התדר הדרוש"** – כותבים $T_{new} = T_{old}/Speedup$ כמכפלת יחסים (IC פי כמה, CPI פי כמה) ומבודדים את CR_{new}. ה-IC מצטמצם.

דוגמה – הטבלה, הזמן וההשוואה

$IC = 10^6$, $CR = 2\,GHz$, שלוש קבוצות פקודות:

קבוצה	w	CPI	$w \cdot CPI$
A	0.5	1	$0.5 \cdot 1 = 0.5$
B	0.3	2	$0.3 \cdot 2 = 0.6$
C	0.2	3	$0.2 \cdot 3 = 0.6$
סה"כ	1		1.7

$$T_1 = \frac{10^6\,[ins] \cdot 1.7\,[cc/ins]}{2 \cdot 10^9\,[cc/sec]} = 850\,\mu s \qquad MIPS_1 = \frac{2 \cdot 10^9}{1.7 \cdot 10^6} \approx 1176.5$$

מכונה 2 – אותו ISA (ולכן אותו IC), $CR = 3\,GHz$, $CPI = 2.4$: $T_2 = \frac{10^6 \cdot 2.4}{3 \cdot 10^9} = 800\,\mu s$. מכונה 2 מהירה פי $\frac{850}{800} = 1.0625$ – למרות ה-CPI הגבוה יותר. ה-MIPS שלה 1250.

התדר הדרוש: מכונה ב-$2\,GHz$ עם $CPI = 1.5$; מהדר חדש מקטין את IC ב-20% ומעלה את ה-CPI ב-50%. כדי להאיץ פי 1.5:

$$\frac{0.8\,IC \cdot 2.25}{CR_{new}} = \frac{1}{1.5} \cdot \frac{IC \cdot 1.5}{2\,GHz} \Rightarrow CR_{new} = 0.8 \cdot 2.25 \cdot 1.5 \cdot \frac{2\,GHz}{1.5} = 3.6\,GHz$$

שימו לב: יחסים במקום מספרים: כשנתון רק ״פי כמה״ השתנו IC ו-CPI, הזמן החדש הוא הזמן הישן כפול היחסים – לא צריך את התדר: מהדר B עם $IC \times 1.4$ ו-$CPI \times 0.8$ רץ $15 \cdot 1.4 \cdot 0.8 = 16.8$ שניות (2022ג, מועד 65, שאלה 7). האפשרות ״חסר תדר״ שגויה. **טבלה בעברית** נקראת **מימין לשמאל**: קריאה הפוכה של טבלת ה-CPI נותנת אפשרות שגויה שמופיעה בין התשובות (2021א, מועד 83, שאלה 3).

1.3 חוק אמדל

תבנית – ״חלק מזמן הריצה מואץ פי S – מה ההאצה הכוללת / מה ה-S הדרוש״

1. F **הוא חלק מהזמן**, לא מהפקודות. נתון אחוז פקודות – מתרגמים קודם לזמן (מחזורים של הקבוצה חלקי כל המחזורים).
2. **טבלת החלקים** – לכל חלק: החלק מהזמן, פי כמה הוא מואץ, והחלק החדש (F/S). החלק שלא שופר נשאר כמו שהוא.
3. **סכום החלקים החדשים** הוא T_{new}/T_{old}, וההאצה היא $1/\Sigma$. שני שיפורים בחלקים נפרדים – שתי שורות באותה טבלה.
4. ״**מה ה-S הדרוש**״ – משוואה אחת: $= \frac{1}{(1-F)+F/S}$ היעד. אם $1 - F$ כבר גדול מ-1/היעד – אין S שמספיק.

דוגמה – 60% מהזמן מואצים פי 4

חלק	חלק מהזמן	האצה	חלק חדש
משופר	0.6	4	$0.6/4 = 0.15$
לא משופר	0.4	1	0.4
סה״כ	1		0.55

$Speedup = \frac{1}{0.55} = 20/11 \approx 1.818$. תוכנית של 100 שניות רצה עכשיו 55 שניות. גם עם $S \to \infty$ ההאצה לא עוברת $\frac{1}{0.4} = 2.5$.

כדי להאיץ פי 2: $\frac{1}{0.4+0.6/S} = 2 \Rightarrow 0.6/S = 0.1 \Rightarrow S = 6$.

שימו לב: מחשבון פשוט בלבד בבחינה: משאירים שברים ($\frac{20}{11}$) עד הסוף ומשווים לאפשרויות רק בשלב האחרון.

1.4 לולאה – כמה איטרציות בזמן נתון

תבנית – ״לולאה של N פקודות רצה T שניות – כמה איטרציות״

1. **CPI ממוצע של גוף הלולאה** מטבלת הקבוצות (1.2).
2. **זמן איטרציה אחת** $= N \cdot CPI \cdot CCT$.
3. **מספר האיטרציות** = הזמן הכולל חלקי זמן איטרציה. ״פקודות בשנייה״ (IPS) הוא CR/CPI.

דוגמה

לולאה של 400 פקודות: 25% lw (CPI 5), 50% פקודות ALU (CPI 4), 25% הסתעפויות (CPI 3), שעון $2\,GHz$, זמן ריצה 2 שניות.
$CPI = 0.25 \cdot 5 + 0.5 \cdot 4 + 0.25 \cdot 3 = 4$; איטרציה אחת: $400 \cdot 4 \cdot 0.5\,ns = 800\,ns$; ובשתי שניות: $\frac{2}{800 \cdot 10^{-9}} = 2.5 \cdot 10^6$ איטרציות.

באילו מבחנים: בכל מבחן, פעמיים. זמן ו-CPI ממוצע (2026ג, מועד 84, שאלה 4; 2026א, מועד 82, שאלה 5; 2025ג, מועד 91, שאלות 4–5)· אמדל (2026ג, מועד 84, שאלה 5; 2026א, מועד 96, שאלה 4; 2026א, מועד 91, שאלה 5; 2022א, מועד 75, שאלה 4)· איטרציות של לולאה (2026א, מועד 96, שאלה 5; 2025א, מועד 94, שאלה 5; 2025א, מועד 81, שאלה 3)· פקודות בשנייה (2026א, מועד 91, שאלה 4; 2024ג, מועד 84, שאלה 5)· שני מהדרים (2025א, מועד 94, שאלה 4; 2024א, מועד 91, שאלה 8; 2022ג, מועד 65, שאלה 7)· איזו נוסחה נכונה (2025א, מועד 81, שאלה 6).

2 ייצוג מספרים

2.1 בסיסים והמרות

ניסוח	כתיב
ערך מיקומי – ספרה a_i במקום i (מימין, מ-0) שווה $a_i \cdot b^i$. ספרות בבסיס b הן $0 \dots b-1$. (מדריך עמ' 25–27)	$(213)_4 = 2 \cdot 16 + 1 \cdot 4 + 3 = 39$ $(52)_7 = 5 \cdot 7 + 2 = 37$
מעשרוני לבסיס b – **סולם החלוקה:** מחלקים ב-b שוב ושוב, ורושמים את השארית. קוראים את השאריות **מלמטה למעלה**. (עמ' 27–28)	n | $n \div 16$ | **שארית** 156 | 9 | C = 12 9 | 0 | 9 $156 = (9C)_{16}$
הקסדצימלי ↔ בינרי – כל ספרה הקסדצימלית היא 4 סיביות (nibble). מקבצים מימין ומשלימים אפסים משמאל. בבחינה ובקוד כותבים `0x` לפני הקסדצימלי. (עמ' 30, 63)	$(9C)_{16} = 1001\ 1100$ $(2D)_{16} = 0010\ 1101 = 45$

2.2 שלוש שיטות ייצוג ב-n סיביות

שיטה	הטווח	ב-8 סיביות	מה לזכור	מדריך
ללא סימן (unsigned)	$0 \dots 2^n - 1$	$0 \dots 255$	כתובות, תווים	עמ' 31–32
גודל וסימן (sign-magnitude)	$-(2^{n-1}-1) \dots 2^{n-1}-1$	$-127 \dots 127$	MSB = סימן, השאר = הגודל. שני אפסים (+0, −0)	עמ' 32
משלים ל-2 (two's complement)	$-2^{n-1} \dots 2^{n-1}-1$	$-128 \dots 127$	**ברירת המחדל** של "עם סימן". אפס אחד; MSB = 1 – שלילי	עמ' 33

טווחי משלים ל-2: 12 סיביות $-2048 \dots 2047$ · 16 סיביות $-32768 \dots 32767$ (שדה ה-immediate) · 32 סיביות $-2^{31} \dots 2^{31}-1$.

ניסוח	כתיב
ערך של מספר במשלים ל-2 – הסיבית העליונה שוקלת **מינוס** 2^{n-1}. (עמ' 35–36)	$A = -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$ $11111010 = -128 + 122 = -6$
היפוך סימן – הופכים כל סיבית ומוסיפים 1. דרך קצרה: מעתיקים מימין עד ה-1 הראשון (כולל), והופכים את כל השאר. (עמ' 33–35)	`0xFA` $6 = 00000110 \to 11111001 + 1 = 11111010 = -6$
הרחבה ל-32 סיביות – **הרחבת סימן** (S.E.) משכפלת את סיבית הסימן; **הרחבת אפסים** (Z.E.) ממלאת אפסים. `lb`, `lh` וה-immediate של `addi`, `lw`, `sw`, `beq` – סימן; `lbu`, `lhu` – אפסים. (עמ' 40, 81)	הבית `0xF0` (MARS): `0x000000F0` ⟶ `lbu` · `0xFFFFFFF0` ⟶ `lb`

2.3 חיבור, חיסור וגלישה

ניסוח	כתיב
חיבור – חיבור בינרי רגיל עמודה אחרי עמודה, עם שורת נשא מעל. במשלים ל-2 **הנשא הסופי נזרק**. **חיסור** – מחברים את המשלים ל-2 של המחסר: $x - y = x + (\overline{y} + 1)$. (עמ' 36–40)	$x - y = x + \overline{y} + 1$
גלישה במשלים ל-2 – רק כששני המחוברים באותו סימן: שני חיוביים נותנים שלילי, או שני שליליים נותנים חיובי. שקול: הנשא **לתוך** ה-MSB שונה מהנשא **החוצה** ממנו. חיובי ועוד שלילי – אף פעם לא גולש. (עמ' 38, 100)	$V = c_{in}(MSB) \oplus c_{out}(MSB)$
גלישה ללא סימן – נשא החוצה מה-MSB ($C = 1$). אותן סיביות, שתי תשובות שונות: הכול תלוי בשיטה שבשאלה. **הכפלה ב-2** (הזזה שמאלה) גולשת: ללא סימן – אם ה-MSB הוא 1; עם סימן – אם שתי הסיביות העליונות שונות. (עמ' 38–39, 46)	$100 + 200$ ב-8 סיביות: 44, $C = 1$ – גלישה ללא סימן אותן סיביות כ-$44 = 100 + (-56)$ – אין גלישה

ניסוח	כתיב
גלישה ב-MIPS – גורמות לחריגה (exception) כשיש גלישה `add`, `addi`, `sub` במשלים ל-2; `addu`, `addiu`, `subu` לא בודקות גלישה. (עמ׳ 82–83)	(MARS): `0x7FFFFFFF + 1` arithmetic overflow → `addi` · `0x80000000` → `addiu`

תבנית – ״חברו / חסרו וקבעו אם הייתה גלישה״

1. **רושמים את שני המספרים ב-n הסיביות של השאלה** (שלילי – היפוך סימן). חיסור – הופכים את סימן המחסר ומחברים.
2. **טבלת עמודות** משמאל לימין b0 … b7: שורת הנשא, שני המחוברים (ובצד הערך העשרוני), והתוצאה.
3. **גלישה:** משווים את הנשא לתוך ה-MSB לנשא החוצה ממנו (או: סימני המחוברים מול סימן התוצאה). ללא סימן – רק הנשא החוצה.
4. **בדיקה בעשרוני:** התוצאה האמיתית בטווח? אם לא – חייבת להיות גלישה.

דוגמה – $50 + 100$ ב-8 סיביות

	b7	b6	b5	b4	b3	b2	b1	b0	
נשא	1	1	0	0	0	0	0	0	
A	0	1	1	0	0	1	0	0	(100)
B	0	0	1	1	0	0	1	0	(50)
A+B	**1**	0	0	1	0	1	1	0	(−106)

התוצאה 10010110. הנשא לתוך b7 הוא 1 והנשא החוצה 0, ולכן $V = 1$: גלישה – הסכום האמיתי 150 גדול מ-127. גם $-100 - 50$ גולש: מתקבל `0x6A` = 106.

2.4 תבנית – ״חשבו בבסיס b וייצגו ב-n סיביות״

תבנית – מספרים בבסיס זר, פעולה, ותוצאה במשלים ל-2 או בגודל וסימן (הקסדצימלי)

1. **כל מחובר לעשרוני** לפי הערך המיקומי (2.1). לא מחשבים בבסיס הזר – שם טועים.
2. **הפעולה בעשרוני**, עם הסימן.
3. **טווח:** התוצאה נכנסת ב-n סיביות בשיטה שבשאלה? אם לא – זו גלישה, ולעיתים זו התשובה.
4. **הערך המוחלט לבינרי** (סולם החלוקה) ומשלימים אפסים משמאל ל-n סיביות.
5. **שלילי:** במשלים ל-2 – הופכים ומוסיפים 1; בגודל וסימן – רק מדליקים את ה-MSB.
6. **מקבצים ל-4** מימין ורושמים הקסדצימלי.

דוגמה – $(23)_6 - (45)_6$ ב-12 סיביות

$(23)_6 = 15$, $(45)_6 = 29$, ולכן ההפרש -14 – בטווח של 12 סיביות. $1110\ 0000\ 0000 = 14$.

שיטה	12 הסיביות	הקסדצימלי
משלים ל-2	1111 1111 0010	0xFF2
גודל וסימן	1000 0000 1110	0x80E

שימו לב:

- **איזו שיטה:** ״גודל וסימן״ ו״משלים ל-2״ נותנים סיביות שונות לאותו מספר, וגם הייצוג השני מופיע בין האפשרויות (2026א, מועד 96, שאלה 3; 2025א, מועד 81, שאלה 4).
- **גלישה היא תשובה:** בחיסור ב-12 סיביות התשובה הנכונה הייתה ״גלישה״; מי שבחר בסיביות של התוצאה בלי לבדוק טווח טעה (2026ג, מועד 84, שאלה 3).
- **ספרה שלא קיימת בבסיס** (למשל 6 בבסיס 6) – סימן שקראתם את הבסיס לא נכון.

באילו מבחנים: בכל מבחן שאלה אחת. בסיס זר וייצוג (2026א, מועד 96, שאלה 3; 2026א, מועד 91, שאלה 3; 2025ג, מועד 91, שאלה 3; 2025א, מועד 94, שאלה 3; 2025א, מועד 81, שאלה 4; 2024ג, מועד 84, שאלה 4; 2024א, מועד 91, שאלה 4; 2023א, מועד 89, שאלה 2) · חיסור וגלישה (2026ג, מועד 84, שאלה 3; 2022א, מועד 75, שאלה 5) · שלילי להקסדצימלי או לבסיס אחר (2026א, מועד 82, שאלה 4; 2023א, מועד 89, שאלה 4; 2023א, מועד 67, שאלה 3; 2022ג, מועד 65, שאלה 3).

3 לוגיקה ורכיבים

3.1 אלגברה בוליאנית ושערים

סדר הפעולות: NOT, אחריו AND, אחריו OR. כל חוק מופיע בזוג – בגרסת OR ובגרסת AND (עקרון הדואליות: מחליפים בין + ל-· ובין 0 ל-1). (מדריך עמ׳ 48, 52)

חוק	בגרסת OR	בגרסת AND
איבר יחידה	$A+0=A$	$A\cdot 1=A$
איון	$A+1=1$	$A\cdot 0=0$
אידמפוטנטיות	$A+A=A$	$A\cdot A=A$
הופכי	$A+\overline{A}=1$	$A\cdot\overline{A}=0$
פילוג	$A+BC=(A+B)(A+C)$	$A(B+C)=AB+AC$
דה-מורגן	$\overline{A+B}=\overline{A}\,\overline{B}$	$\overline{AB}=\overline{A}+\overline{B}$
צמצום	$A+AB=A$	$A(A+B)=A$
ההופכי הנעלם	$A+\overline{A}B=A+B$	$A(\overline{A}+B)=AB$

חילוף וקיבוץ מתקיימים לשתי הפעולות, ו-$\overline{\overline{A}}=A$.

ניסוח	כתיב
צורות קנוניות – לפונקציה של n משתנים יש 2^n שורות בטבלת האמת. **סכום מכפלות** (SOP) – סכום ה-minterms של השורות שבהן $F=1$; **מכפלת סכומים** (POS) – מכפלת ה-maxterms של השורות שבהן $F=0$. מספר השורה הוא הערך הבינרי של הכניסות. (עמ׳ 49–52)	$F=\overline{A}B+AC$ $=\Sigma m(2,3,5,7)=\Pi M(0,1,4,6)$
כמה פונקציות – כל עמודת פלט אפשרית בטבלה של 2^n שורות היא פונקציה אחרת. (עמ׳ 49)	2^{2^n}: 16 פונקציות של 2 משתנים, 256 של 3

שער	הביטוי	הפלט בשורות $xy=00,01,10,11$	הערה
AND	xy	0001	
OR	$x+y$	0111	
NAND	$\overline{xy}$	1110	אוניברסלי לבדו
NOR	$\overline{x+y}$	1000	אוניברסלי לבדו
XOR	$x\oplus y=x\overline{y}+\overline{x}y$	0110	1 כשהכניסות שונות; $x\oplus 1=\overline{x}$
XNOR	$\overline{x\oplus y}=xy+\overline{x}\,\overline{y}$	1001	1 כשהכניסות שוות

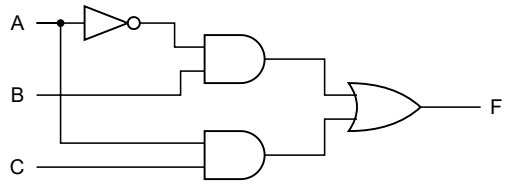
עיגול קטן ביציאה (או בכניסה) של שער הוא NOT. כל פונקציה ניתנת לבנייה מ-AND, OR ו-NOT, וגם משערי NAND בלבד או משערי NOR בלבד. (עמ׳ 53–54, 97)

3.2 תבנית – ״מהו המוצא F של המעגל / איזה ביטוי שקול״

תבנית – מהמעגל לביטוי

1. **רושמים על כל חוט את הביטוי שלו**, משער לשער לפי כיוון הזרימה. שער עם עיגול ביציאה (NAND, NOR) – מוסיפים את ההיפוך.
2. **טבלת אמת עם עמודה לכל שער**: הכניסות, עמודה לכל ביטוי ביניים לפי סדר הזרימה, ו-F. שמונה שורות בסדר 000 … 111.
3. **בודקים כל אפשרות** – עמודה משלה ליד F; נכונה רק אפשרות שזהה ל-F בכל השורות.
4. ״**הביטוי הפשוט ביותר**״ – כמה אפשרויות יכולות להיות שוות ל-F. בוחרים את הקצרה, ומראים שרשרת אלגברית: צעד בשורה, והחוק בצד.

דוגמה – $F = \overline{A}B + AC$, **מול האפשרות** $(A+B)(\overline{A}+C)$



A	**B**	**C**	$\overline{A}$	$\overline{A}B$	AC	**F**	$(A+B)(\overline{A}+C)$
0	0	0	1	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	1	0	1	1
0	1	1	1	1	0	1	1
1	0	0	0	0	0	0	0
1	0	1	0	0	1	1	1
1	1	0	0	0	0	0	0
1	1	1	0	0	1	1	1

העמודות זהות, ולכן האפשרות שקולה ל-F.

שרשרת אלגברית (דוגמה נוספת):

$$AB + A\overline{B} + \overline{A}B \;=\; A(B+\overline{B}) + \overline{A}B \;=\; A + \overline{A}B \;=\; A + B$$

פילוג, הופכי ויחידה, ההופכי הנעלם.

שימו לב:

- **השער האחרון:** מי שעצר ב-AND שלפני ה-NAND האחרון קיבל 0 במקום 1 (2025א, מועד 81, שאלה 2). מסיימים את הטבלה עד היציאה.
- **שתי אפשרויות נכונות:** כשגם הביטוי המלא וגם המצומצם מופיעים, והשאלה מבקשת את **הפשוט** – רק המצומצם (2025ג, מועד 91, שאלה 2).

3.3 רכיבים צירופיים: מפענח, מרבב, מחבר

ניסוח	כתיב
מפענח (decoder) – $n \to 2^n$ – בדיוק יציאה אחת היא 1: זו שהאינדקס שלה הוא הערך הבינרי של הכניסות. עם כניסת אפשור (Enable) – כש-Enable = 0 כל היציאות 0. **מקודד עדיפויות** – הפוך: מחזיר את האינדקס של הכניסה הדולקת בעלת העדיפות הגבוהה. (עמ' 92–94)	מפענח 3→8: $D_5 = A\overline{B}C$
מרבב (multiplexer) – $n - 2^n \to 1$ קווי בורר; היציאה היא הכניסה שהאינדקס שלה הוא ערך הבורר. מרבב של מילים (32 סיביות) הוא 32 מרבבים של סיבית אחת. (עמ' 94–95)	$out = \overline{S}\,I_0 + S\,I_1$
מחבר – חצי מחבר: $a \oplus b$ ונשא ab. **מחבר מלא** (FA) מוסיף נשא נכנס. מחבר אדווה (ripple carry) של n סיביות: הנשא עובר מ-FA ל-FA, ולכן הזמן $n \cdot \Delta T$. (עמ' 96–97)	$Sum = a \oplus b \oplus c_{in}$ $c_{out} = ab + a\,c_{in} + b\,c_{in}$

a	0	0	0	0	1	1	1	1
b	0	0	1	1	0	0	1	1
c_{in}	0	1	0	1	0	1	0	1
Sum	0	1	1	0	1	0	0	1
c_{out}	0	0	0	1	0	1	1	1

תבנית – "המרבב מממש פונקציה F"

1. **טבלה:** הכניסות, ערך הבורר בכל שורה, **איזו כניסה נבחרה**, וערכה – זה F.
2. **מרבבים בשרשרת** – קודם עמודה ליציאה של המרבב הראשון, ואז היא כניסה של השני.
3. **ממזערים** את F לפי שורות ה-1 (Σm) והאלגברה של 3.1, ומשווים לאפשרויות.

דוגמה – מרבב 4→1 עם $S_1 = A$, $S_0 = B$ **וכניסות** $I_0 = C$, $I_1 = \overline{C}$, $I_2 = 0$, $I_3 = 1$

A	B	C	S_1S_0	נבחר	F
0	0	0	00	$I_0 = C$	0
0	0	1	00	$I_0 = C$	1
0	1	0	01	$I_1 = \overline{C}$	1
0	1	1	01	$I_1 = \overline{C}$	0
1	0	0	10	$I_2 = 0$	0
1	0	1	10	$I_2 = 0$	0
1	1	0	11	$I_3 = 1$	1
1	1	1	11	$I_3 = 1$	1

$F = \Sigma m(1,2,6,7) = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB$.

3.4 ה-ALU

ניסוח	כתיב
יחידת ALU של סיבית אחת – כניסות a, b, נשא נכנס, Less; קווי בקרה Ainvert, Bnegate ו-Operation (2 סיביות): 00 – AND, 01 – OR, 10 – חיבור, 11 – Less. 32 יחידות בשרשרת: הנשא היוצא של סיבית i הוא הנכנס של $i+1$. (עמ' 96–98)	$a - b = a + \overline{b} + 1$: Bnegate הופך את b ונותן נשא נכנס 1 לסיבית 0
slt – מחסרים $a - b$. יחידת ה-MSB מוציאה **Set** = סיבית הסכום שלה, והוא חוזר לכניסת Less של סיבית 0; Less של כל השאר 0. Ainvert – **NOR** = Bnegate = 1 ופעולת AND (דה-מורגן). (עמ' 98–99)	$\overline{a} \cdot \overline{b} = \overline{a + b}$

ALU control (Ainvert, Bnegate, Operation)	0000	0001	0010	0110	0111	1100
הפעולה	and	or	add	sub	slt	nor

(עמ' 99; 1101 – NAND)

ניסוח	כתיב
ארבעת החיווים – **Z** (Zero): כל 32 סיביות התוצאה 0 – NOR של כולן; beq מחסרת ובודקת את **S**. Z: סיבית 31 של התוצאה. **O** (Overflow): גלישה במשלים ל-2, $c_{in31} \oplus c_{out31}$. **C** (Carry): הנשא היוצא מסיבית 31 – גלישה ללא סימן. (עמ' 99–100)	$O = c_{in31} \oplus c_{out31}$ $Z = \overline{R_{31} + \cdots + R_1 + R_0}$

תבנית – ״מה ערכי Zero / Overflow / Set״ או ״מה יודעת יחידת ה-MSB״

1. **מחשבים את התוצאה ב-32 סיביות** (הקסדצימלי, נשא סופי נזרק). חיסור – $a + \overline{b} + 1$.
2. **Z** – האם כל התוצאה אפס; **S** – הסיבית העליונה; **O** – כלל הסימנים של 2.3; **C** – הנשא החוצה.
3. **יחידת ה-MSB לבדה** רואה רק את a_{31}, b_{31}, הנשא הנכנס שלה וקווי הבקרה: Result31, Set ו-Overflow נקבעים בה – **Zero לא** (תלוי בכל הסיביות).

דוגמה – ה-ALU בנוי מ-32 יחידות של סיבית אחת (חושב בסימולציה של היחידות)

פעולה	a	b	התוצאה	Z	S	O	C
add	0x7FFFFFFF	0x00000001	0x80000000	0	1	1	0
add	0x08000000	0xF8000000	0x00000000	1	0	0	1
sub	0x00000005	0x00000007	0xFFFFFFFE	0	1	0	0
sub (beq)	0x00000007	0x00000007	0x00000000	1	0	0	1

שימו לב:

- **0 בגלל גלישה מסביב:** `0xF8000000 + 0x08000000` נותן `0x1_00000000`, כלומר 0 ב-32 סיביות – ולכן Zero = 1, ואין גלישה (חיובי ועוד שלילי). המפתח הרשמי סימן אחרת וטעה (2022א, מועד 82, שאלה 7).
- **Set לבדו שגוי כשיש גלישה:** $a =$ `0x7FFFFFFF`, $b = -1$: החיסור גולש, Set = 1 אף ש-$a > b$. התיקון: $Set \oplus Overflow$ (עמ׳ 100, תרגיל B.24 בספר).
- **״מה לא ניתן לחשב ביחידת ה-MSB״:** Zero (2025א, מועד 87, שאלה 5; 2022ג, מועד 92, שאלה 4).

3.5 לוגיקה סדרתית ומקבץ האוגרים

ניסוח	כתיב
מנעול SR (שני שערי NOR במשוב) – $S = 1$ מדליק את Q, $R = 1$ מכבה, $S = R = 0$ שומר, $S = R = 1$ אסור. **מנעול D** – כש-$C = 1$ הוא **שקוף**: Q עוקב אחרי D כל הזמן; כש-$C = 0$ – שומר. (עמ׳ 107–108)	מנעול D: $S = C \cdot D$, $R = C \cdot \overline{D}$
דלגלג (flip-flop) **D** – שני מנעולים, ראשי ומשני, שאחד מהם תמיד נעול. Q משתנה **רק בקצה** של השעון: בתרשים B.8.4 בקצה היורד (1→0), ומקבל את ערך D שלפני הקצה. D צריך להיות יציב זמן setup לפני הקצה וזמן hold אחריו. (עמ׳ 106, 109)	ראשי פתוח כש-$C = 1$, משני פתוח כש-$C = 0$
מקבץ האוגרים – 32 אוגרים של 32 סיביות = 1024 דלגלגי D. **קריאה:** שני מרבבים של 32→1 (כל אחד ברוחב 32 סיביות), בוררים rs ו-rt (5 סיביות כל אחד); אין שעון בקריאה. **כתיבה:** מפענח 5→32 על מספר האוגר, AND עם **RegWrite**, לכניסת השעון של האוגר; נכתב בסוף מחזור השעון. (עמ׳ 109–111)	64 אוגרים: שדות של 6 סיביות, מרבבים 64→1, מפענח 6→64

תבנית – ״מה ערך Q אחרי צורת הגל״

1. **טבלת זמן:** שורות C ו-D, עמודה לכל צעד זמן; Q מתחיל בערך הנתון.
2. **מנעול:** בעמודה שבה $C = 1$, ערך Q הוא ה-D של אותה עמודה; בעמודה שבה $C = 0$, ערך Q כמו בעמודה הקודמת.
3. **דלגלג:** Q משתנה רק בעמודה של הקצה (יורד בתרשים B.8.4; אם השאלה אומרת ״עולה״ – שם), ומקבל את D מהעמודה שלפני הקצה.

דוגמה – מנעולי NOR בסימולציה ברמת השערים, Q מתחיל ב-0

t	0	1	2	3	4	5	6	7	8	9
C	0	1	1	1	0	0	1	1	0	0
D	0	1	0	1	1	0	0	0	1	1
מנעול Q	0	1	0	1	1	1	0	0	0	0
דלגלג Q	0	0	0	0	1	1	1	1	0	0

המנעול עוקב אחרי כל שינוי של D בזמן ש-$C = 1$ (צעדים 1–3). הדלגלג משתנה רק בקצוות היורדים, בצעדים 4 ו-8, ומקבל את D של צעדים 3 ו-7.

באילו מבחנים: שתיים–שלוש השאלות הראשונות בכל מבחן. מעגל וביטוי (2026ג, מועד 84, שאלות 1–2; 2026א, מועד 96, שאלות 1–2; 2026א, מועד 82, שאלות 1–2; 2025ג, מועד 91, שאלות 1–2; 2025א, מועד 94, שאלות 1–2; 2025א, מועד 87, שאלות 1–2; 2025א, מועד 81, שאלות 1–2; 2023א, מועד 89, שאלות 1, 3) · מספר פונקציות (2024א, מועד 91, שאלה 3; 2022ג, מועד 65, שאלה 4) · מרבבים (2026א, מועד 91, שאלה 2; 2024ג, מועד 84, שאלות 1, 3; 2022ג, מועד 65, שאלה 1) · ALU (2025א, מועד 87, שאלה 5; 2025א, מועד 81, שאלה 7; 2022ג, מועד 92, שאלה 4; 2022א, מועד 82, שאלה 7) · מנעולים ודלגלגים (2026א, מועד 82, שאלה 3; 2024ג, מועד 84, שאלה 7) · מקבץ של 64 אוגרים (2024א, מועד 91, שאלה 11).

4 שפת אסמבלי MIPS

4.1 אוגרים, זיכרון ופורמטים

מספר	שם	תפקיד	נשמר בקריאה לפרוצדורה?
0	`$zero`	הקבוע 0 – כתיבה אליו לא משנה אותו	–
1	`$at`	שמור לאסמבלר (לפסאודו-פקודות)	–
3–2	`$v1`–`$v0`	ערכי החזרה; `$v0` הוא גם קוד ה-syscall	לא
7–4	`$a3`–`$a0`	ארבעת הארגומנטים	לא
25–24, 15–8	`$t9`–`$t0`	זמניים	לא – הקורא שומר אם צריך
23–16	`$s7`–`$s0`	שמורים	**כן** – פרוצדורה שמשנה אותם שומרת ומשחזרת
30, 29, 28	`$fp`, `$sp`, `$gp`	מצביע גלובלי, ראש המחסנית, מסגרת	כן
31	`$ra`	כתובת החזרה ש-`jal` כותבת	כן (בפרוצדורה שקוראת לאחרת)

(מדריך עמ' 61, 70–73; 26–27 `$k0`, `$k1` – למערכת ההפעלה)

ניסוח	כתיב
זיכרון – כתובת לכל בית; מילה = 4 בתים, וכתובת של מילה (`lw`, `sw`) חייבת להתחלק ב-4. כתובת הגישה היא האוגר ועוד ההיסט **בהרחבת סימן**. (עמ' 61–63)	`lw $t0,-8($s6)`: `$t0` ← MEM[`$s6` − 8]
סדר הבתים – big endian: הבית **העליון** של המילה בכתובת הנמוכה (כך בספר); little endian: הבית התחתון בכתובת הנמוכה (כך MARS על מחשב PC). (עמ' 63; MARS)	המילה `0x12345678` בכתובת `0x10010000`: ב-MARS `lbu` מהכתובת הזאת נותן `0x78`
מפת הזיכרון ב-MARS – קוד מ-`0x00400000`, נתונים (`.data`) מ-`0x10010000`, `$sp` מתחיל ב-`0x7fffeffc` והמחסנית גדלה **כלפי מטה**. (MARS; עמ' 71–72)	`.text 0x00400000` · `.data 0x10010000`

פורמט	31–26	25–21	20–16	15–11	10–6	5–0
R	op (6) = 0	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
I	op (6)	rs (5)	rt (5)	immediate (16)		
J	op (6)	address (26)				

(עמ' 65–66, 78) ב-R: `add $rd,$rs,$rt` – היעד **שלישי בקידוד** אבל ראשון בכתיבה. ב-I: `addi`/`lw` כותבות ל-rt. **שיטות מיעון:** אוגר (`add`), בסיס+היסט (`lw`), מיידי (`addi`), יחסי ל-PC (`beq`), פסאודו-ישיר (`j`).

4.2 הפקודות ופסאודו-הפקודות

הקודים לפי MARS (op לפקודות I/J, funct לפקודות R, שה-op שלהן 0). (עמ' 67–70, 78, 81–84, 119–129)

פקודה	קוד	מה היא עושה	שימו לב
`add` / `addu` / `sub` / `subu`	funct `0x20` / `0x21` / `0x22` / `0x23`	`$rd = $rs ± $rt`	רק `add`, `sub` (ו-`addi`) עוצרות בגלישה
`addi` / `addiu`	op `0x08` / `0x09`	`$rt = $rs + SE(imm)`	ההיסט בהרחבת סימן
`and` / `or` / `xor` / `nor`	funct `0x24` / `0x25` / `0x26` / `0x27`	פעולה סיבית בין שני אוגרים	NOT = `nor $t0,$t1,$zero`
`andi` / `ori`	op `0x0c` / `0x0d`	`$rt = $rs and/or ZE(imm)`	**הרחבת אפסים**, לא סימן
`sll` / `srl` / `sra`	funct `0x00` / `0x02` / `0x03`	הזזה ב-`shamt` מקומות	`srl` ממלא 0, `sra` משכפל את סיבית הסימן
`slt` / `sltu`	funct `0x2a` / `0x2b`	`$rd = 1` אם `$rs < $rt`, אחרת 0	`slt` עם סימן, `sltu` בלי
`slti` / `sltiu`	op `0x0a` / `0x0b`	`$rt = 1` אם `$rs < SE(imm)`	
`lui`	`0x0f` op	`$rt = imm << 16`	הסיביות התחתונות 16 0
`lw` / `sw`	op `0x23` / `0x2b`	מילה מ/אל `$rs + SE(offset)`	כתובת שמתחלקת ב-4
`lb` / `lbu` / `sb`	op `0x20` / `0x24` / `0x28`	בית אחד	`lb` הרחבת סימן, `lbu` אפסים

פקודה	קוד	מה היא עושה	שימו לב
`lh` / `lhu` / `sh`	op `0x21` / `0x25` / `0x29`	חצי מילה	כמו בבית
`beq` / `bne`	op `0x04` / `0x05`	קפיצה אם שווים / שונים	יעד יחסי ל-PC+4 (4.6)
`j` / `jal`	op `0x02` / `0x03`	קפיצה; `jal` גם `$ra = PC+4`	26 סיביות כתובת
`jr`	`0x08` funct	`PC = $rs`	`jr $ra` – חזרה מפרוצדורה
`mult` / `multu`	funct `0x18` / `0x19`	מכפלה של 64 סיביות ב-Hi:Lo	לא במקבץ האוגרים Hi, Lo
`div` / `divu`	funct `0x1a` / `0x1b`	Lo = המנה, Hi = השארית	`0x12 mflo`, `0x10 mfhi` מעתיקות לאוגר
`syscall`	`0x0c` funct	שירות לפי `$v0`	1 הדפסת שלם, 4 מחרוזת, 5 קריאת שלם, 10 סיום, 11 תו

פסאודו-פקודות – מותרות בקוד, והאסמבלר מתרגם אותן לפקודות אמיתיות דרך `$at` (MARS): (עמ' 78, 80)

פסאודו	התרגום	הערה
`li $t0,5`	`addiu $t0,$zero,5`	קבוע קטן – פקודה אחת
`li $t0,0x12345678`	`lui $at,0x1234` `ori $t0,$at,0x5678`	קבוע של 32 סיביות – שתיים
`la $t0,lab`	`lui $at,0x1001` `ori $t0,$at,0`	כתובת התווית (0x10010000 = lab)
`move $t0,$t1`	`addu $t0,$zero,$t1`	
`blt $t0,$t1,L`	`slt $at,$t0,$t1` `bne $at,$zero,L`	קופצת אם $t0 < t1$
`bgt $t0,$t1,L`	`slt $at,$t1,$t0` `bne $at,$zero,L`	האוגרים **מתחלפים**
`ble $t0,$t1,L`	`slt $at,$t1,$t0` `beq $at,$zero,L`	קופצת אם לא $t0 < t1$
`bge $t0,$t1,L`	`slt $at,$t0,$t1` `beq $at,$zero,L`	קופצת אם לא $t0 < t1$

4.3 כתיבת קוד (B1)

תבנית – "כתבו פרוצדורה / תרגמו קטע C"

1. **מפת אוגרים:** טבלה של משתנה | אוגר | תפקיד (כתובת בסיס, מונה, סכום, קבוע). ארגומנטים ב-`$a0`–`$a3`, ערך מוחזר ב-`$v0`.
2. **השלד:** אתחול → **בדיקה בראש הלולאה** (`slt` + `beq` ל-done) → גוף → קידום → `j loop`. הסתעפות על התנאי ההפוך: `if-else` – אל ה-`else`, ו-`j` מעל ה-`else` בסוף ה-`then`.
3. **גישה למערך:** מילים – `sll` ב-2 (כפול 4) ואז `add` לבסיס, `lw`; בתים ומחרוזות – `lbu`/`lb` וקידום ב-1; סוף מחרוזת – הבית 0 (`beq $t1,$zero,done`).
4. **פרוצדורה:** עלה (לא קוראת לאחרת) – רק `$t`, בלי מחסנית, ובסוף `jr $ra`. משתמשת ב-`$s` או קוראת לאחרת – מסגרת (4.5).
5. **הערות** על כל שורה, ומעקב (4.4) על קלט קטן ועל קצוות: ריק, 0, שלילי.

דוגמה – `for (i = 0; i < n; i++) sum += A[i];` ;MARS: $A = \{3, -1, 4\}$ נותן 6

```
loop: slt  $t1,$t0,$s1      # i < n ?
      beq  $t1,$zero,done
      sll  $t2,$t0,2        # 4*i
      add  $t2,$t2,$s0      # &A[i]
      lw   $t3,0($t2)
      add  $s2,$s2,$t3      # sum += A[i]
      addi $t0,$t0,1
      j    loop
done:
```

משתנה	אוגר	תפקיד
A	`$s0`	כתובת הבסיס
n	`$s1`	גבול
i	`$t0`	מונה
sum	`$s2`	הסכום

שימו לב:

- **הסתעפות פסאודו:** `ble`/`bge` הן `slt` + `beq` – בודקים מי משווה למי. בשאלה שביקשה תרגום של `bge` אף אפשרות לא הייתה נכונה; אחת קפצה ב-$\leq$ (2026א, מועד 91, שאלה 8).
- **שוויון:** "גדול" מול "גדול או שווה" בתנאי הלולאה עלה 2 נקודות בכל סעיף (2024א, מועד 91, שאלה 16).

4.4 מעקב – "מה הקוד עושה" (B2)

תבנית – מעקב אחרי קוד

1. **מצב התחלתי:** האוגרים והזיכרון שנתונים (ומה ש-`li`/`la` טוענות). **ערכים בבסיס של הקבועים** – הקסדצימלי נשאר הקסדצימלי.
2. **טבלת מעקב:** שורה לכל **סיבוב** של הלולאה, עמודה לכל אוגר שמשתנה, עמודה "קופצת?", ועמודה לפלט או לתא זיכרון שנכתב.
3. **עד היציאה:** תנאי הלולאה נבדק בכל סיבוב; סופרים כמה פעמים כל פקודה בוצעה אם שואלים על מספר פקודות.
4. **המשמעות:** משפט אחד שמתאר את **כל** האלגוריתם (מה מחושב, באיזה סדר, מה מוחזר) – לא תרגום שורה-שורה.
5. **כל קלט שנשאל** – גם 0, שלילי ומחרוזת ריקה: לולאה שלא נגמרת, או `addi` שגולשת וגורמת לחריגה.

דוגמה – הלולאה רצה כל עוד `$t0 < 20` (MARS: `$t0 = 0x20, $t1 = 5, $t2 = 0`)

```
      li   $t0,1
      li   $t1,0
loop: sll  $t0,$t0,1
      addi $t1,$t1,1
      slti $t2,$t0,20
      bne  $t2,$zero,loop
```

סיבוב	$t0	$t1	$t2	bne
1	2	1	1	קופצת
2	4	2	1	קופצת
3	8	3	1	קופצת
4	16	4	1	קופצת
5	**32**	**5**	0	לא

המשמעות: `$t1` סופר כמה הכפלות ב-2 צריך עד ש-`$t0` מגיע ל-20 לפחות.

ניבים שחוזרים במבחנים (כל אחד הורץ ב-MARS):

הקוד	מה הוא עושה	דוגמה

`sra $t1,$t0,31`
`xor $t2,$t0,$t1`
`subu $t2,$t2,$t1`
	ערך מוחלט: המסכה היא 0 או 0xFFFFFFFF לפי הסימן	$5 \to 5$, $-7 \to 7$

`addi $t1,$t0,-1` `and $t0,$t0,$t1`	מכבה את ה-1 הימני ביותר; בלולאה – סופרת סיביות 1	`0x58` ⟶ `0x50`; ל-`0x58` שלוש סיביות 1
`andi $t1,$t0,1` `srl $t0,$t0,1`	הסיבית הנמוכה (זוגי / אי-זוגי), ומעבר לבאה	1 ⟶ `0x59`
`sll $t1,$t0,2`	כפול 4: אינדקס של מילה ⟶ היסט בבתים	7 ⟶ 28
`srl` מול `sra` ב-1	חלוקה ב-2: `srl` ללא סימן, `sra` עם	-8: `0x7FFFFFFC` מול `0xFFFFFFFC`

שימו לב:

- **שורה אחרי שורה זה לא תשובה:** תיאור חלקי של "מה הפרוצדורה עושה" עלה 2 מתוך 6 נקודות; תשובה מלאה ציינה את כלל הסימן, את החלוקה ב-10 עם דחיפת הספרות ואת ההדפסה בסדר הפוך (2025א, מועד 94, שאלה 12).
- **קצוות:** "$a1 = 0$" – הלולאה לא נגמרת" (2023א, מועד 89, שאלה 16); מילה 0 שתוקעת את הלולאה עד ש-`addi` גולשת (2024ג, מועד 84, שאלה 16). אוגר שמחזיק **כתובת** ולא ערך (2026ג, מועד 84, שאלה 14).

4.5 פרוצדורות ומחסנית (B3)

ניסוח	קוד

קריאה וחזרה – `jal L`: `$ra = PC+4` וקפיצה ל-L. `jr $ra`: חזרה. פרוצדורה שקוראת לאחרת **דורסת** את `$ra` שלה – לכן דוחפת אותו למחסנית קודם. (עמ' 71, 89)	`jal proc  # $ra = PC+4`
	`...`
	`proc: ...`
	`jr $ra`

דחיפה ושליפה – המחסנית גדלה כלפי **מטה**: דחיפה מקטינה את `$sp` ואז כותבת; שליפה קוראת ואז מגדילה. שליפה לא מוחקת – הערך נשאר בזיכרון מתחת ל-`$sp`. (עמ' 71–72)	`addi $sp,$sp,-4  # push $s5`
	`sw $s5,0($sp)`
	`lw $t2,0($sp)  # pop $t2`
	`addi $sp,$sp,4`

ניסוח	קוד

מוסכמת השמירה – $s0–$s7, $ra ו-$sp חייבים לחזור לקורא כמו שהיו; $t, $a, $v – לא. בתחילת הפרוצדורה (prologue) מקצים מסגרת ושומרים, ובסוף (epilogue) משחזרים ומשחררים, באותו סדר הפוך. (עמ' 72–73)	`addi $sp,$sp,-12  # prologue`
	`sw $ra,8($sp)`
	`sw $s1,4($sp)`
	`sw $s0,0($sp)`
	`...`
	`lw $s0,0($sp)  # epilogue`
	`lw $s1,4($sp)`
	`lw $ra,8($sp)`
	`addi $sp,$sp,12`
	`jr $ra`

תבנית – "מה במחסנית / אילו אוגרים צריך לשמור / מעקב אחרי רקורסיה"

1. **אילו אוגרים לשמור:** כל $s שהפרוצדורה כותבת אליו, ו-$ra אם יש בה jal. אוגר $t או $a שנדרש אחרי jal – שומרים אותו לפני הקריאה.
2. **טבלת קריאות:** שורה לכל קריאה – הארגומנט, $sp אחרי הדחיפה, ה-$ra שנשמר והערך המוחזר. $ra של הקריאה הראשונה = הכתובת שאחרי ה-jal בקורא; של הקריאות הרקורסיביות = הכתובת שאחרי ה-jal שבתוך הפרוצדורה.
3. **ציור המחסנית:** כתובת גבוהה למעלה, תוכן, של איזו מסגרת; מסמנים לאן $sp מצביע.
4. **בדיקה:** בסוף $sp חוזר לערכו ההתחלתי, ומספר הדחיפות = מספר השליפות בכל מסלול.

דוגמה – `sum(n) = n + sum(n-1)`, `sum(0) = 0`, נקראת עם n = 3 (MARS, הזיכרון אחרי הריצה)

```
main: li   $a0,3
      jal  sum              # 0x00400004, $ra = 0x00400008
      ...
sum:  addi $sp,$sp,-8       # push a frame of 2 words
      sw   $ra,4($sp)
      sw   $a0,0($sp)
      bne  $a0,$zero,rec
      li   $v0,0            # sum(0) = 0
      addi $sp,$sp,8
      jr   $ra
rec:  addi $a0,$a0,-1
      jal  sum              # 0x00400034, $ra = 0x00400038
      lw   $a0,0($sp)
      lw   $ra,4($sp)
      add  $v0,$v0,$a0
      addi $sp,$sp,8
      jr   $ra
```

מסגרת	תוכן	כתובת
sum(3)	0x00400008 ($ra)	0x7fffeff8
sum(3)	3 ($a0)	0x7fffeff4
sum(2)	0x00400038	0x7fffeff0
sum(2)	2	0x7fffefec
sum(1)	0x00400038	0x7fffefe8
sum(1)	1	0x7fffefe4
sum(0)	0x00400038	0x7fffefe0
sum(0)	0	0x7fffefdc ← $sp

קריאה	$a0	$sp אחרי הדחיפה	$ra שנשמר	מחזירה
sum(3)	3	0x7fffeff4	0x00400008	6
sum(2)	2	0x7fffefec	0x00400038	3
sum(1)	1	0x7fffefe4	0x00400038	1
sum(0)	0	0x7fffefdc	0x00400038	0

$sp מסומן ברגע העמוק ביותר; אחרי החזרה הוא שוב 0x7fffeffc, והמילים נשארות בזיכרון.

שימו לב:

- **$s בלי שמירה:** "הפרוצדורה משתמשת ב-$s בלי לשמור אותם; צריך לדחוף אותם בכניסה ולשלוף ביציאה" – זו התשובה המלאה (2026ג, מועד 84, שאלה 14).
- **סדר:** ב-push מקטינים את $sp **לפני** הכתיבה; כתיבה ב-0($sp) לפני ההקטנה דורסת את המילה של הקורא.

4.6 קידוד לשפת מכונה (B4)

ניסוח	כתיב
יעד של bne/beq – ההיסט נמדד **במילים** מ-PC+4, בהרחבת סימן. טווח: $-2^{15} \dots 2^{15}-1$ מילים מ-PC+4, כלומר $-131068 \cdots + 131072$ בתים מה-beq עצמו. (עמ' 123, 134)	$target = PC + 4 + SE(offset) \cdot 4$ $offset = \frac{target-(PC+4)}{4}$
יעד של jal/j – 26 הסיביות הן כתובת **מילה**: מוסיפים 00 מימין, ומשמאל את 4 הסיביות העליונות של PC+4. אפשר לקפוץ רק בתוך אותו מקטע של 256MB. (עמ' 78, 129–130)	$target = (PC+4)[31:28] \parallel address \parallel 00$
קבוע של 32 סיביות – lui לחצי העליון ואחריו ori לתחתון. la ו-li גדול מתורגמות כך. (עמ' 78, 80)	la $t0,lab+4 מתורגמת ל: lui $at,0x1001 ori $t0,$at,4

תבנית – מאסמבלי לקוד מכונה ובחזרה

1. **הפורמט לפי ה-op:** op = 0 – R (הפעולה ב-funct); jal/j – J (2, 3); כל השאר – I.
2. **טבלת שדות** בעשרוני: op | rs | rt | rd | shamt | funct (או immediate או address). מספרי אוגרים מטבלה 4.1; ב-R היעד הוא rd, ב-I הוא rt.
3. **לבינרי לפי רוחב השדות** (6/5/5/5/5/6 או 6/5/5/16), מחברים ל-32 סיביות, מקבצים ל-4 וקוראים הקסדצימלי.
4. **הסתעפות:** כתובת התווית מהרשימה (הפקודה ה-k נמצאת ב-0x00400000 + $4k$), ואז $(target - (PC+4))/4$; שלילי – משלים ל-2 ב-16 סיביות.
5. **פענוח** (הקסדצימלי → פקודה): פורשים ל-32 סיביות, חותכים לפי ה-op, וה-immediate עובר **הרחבת סימן** (פרט ל-ori/andi).

דוגמה – קידוד (encode.py, זהה ל-MARS)

הפקודה	op	rs	rt	rd	shamt	funct / imm	hex
sub $s0,$t1,$t2	0	9	10	16	0	34	0x012A8022
sw $t3,8($sp)	43	29	11			8	0xAFAB0008
beq $t0,$zero,L	4	8	0			4	0x11000004

- sub בבינרי לפי שדות: 100010 00000 01010 10000 00000 000000.
- beq בכתובת 0x00400010, והתווית L בכתובת 0x00400024: $(0x00400024 - 0x00400014)/4 = 4$. אחורה, מ-0x00400020 אל 0x00400004: ההיסט -8 = 0xFFF8.
- j ב-0x00400030 אל 0x00400008: שדה הכתובת 0x0100002, המילה 0x08100002, והיעד 00||0000010000000000000000010||0000.
- פענוח: 0xAD25FF00 הוא sw $a1,-256($t1) – ה-immediate 0xFF00 עובר הרחבת סימן ל-0xFFFFFF00.

שימו לב:

- **הרחבת סימן:** ב-sw עם immediate 0xFF00 הערך שנכנס ל-ALU הוא -256, לא 65280 (2024ג, מועד 84, שאלה 11).
- **הקסדצימלי עם 0x:** תשובה בעשרוני כשביקשו הקסדצימלי ירדה בנקודות (2024ג, מועד 84).
- **הטווח בבתים מול במילים:** אפשרות שהחליפה בין מספר הבתים קדימה ואחורה הייתה שגויה (2026א, מועד 96, שאלה 7).
- **j למקטע אחר:** 4 הסיביות העליונות באות מ-PC+4, ולכן היעד חייב להיות באותו מקטע (2025ג, מועד 91, שאלה 14).

באילו מבחנים: בכל מבחן כמה שאלות, ושאלה פתוחה של מעקב אחרי פרוצדורה (14–18 נקודות) בכל מבחן מאז 2022. כתיבת קוד (2026ג, מועד 84, שאלה 6; 2026א, מועד 91, שאלה 8; 2023א, מועד 67, שאלה 16; 2022א, מועד 82, שאלה 13; 2022א, מועד 75, שאלה 13; 2021ג, מועד 83, שאלה 15) · מעקב (2026ג, מועד 84, שאלה 7; 2026א, מועד 96, שאלות 6, 14; 2026א, מועד 82, שאלות 6, 14; 2025ג, מועד 91, שאלות 6, 12; 2025א, מועד 87, שאלות 8–9, 12; 2025א, מועד 81, שאלות 8, 12; 2024ג, מועד 84, שאלות 12, 14, 16; 2024א, מועד 91, שאלות 6–7, 16) · פרוצדורות ומחסנית (2026ג, מועד 84, שאלה 14; 2026א, מועד 82, שאלה 7; 2025א, מועד 94, שאלות 10, 12; 2024ג, מועד 84, שאלה 13; 2024א, מועד 91, שאלה 13; 2023א, מועד 89, שאלה 9) · קידוד (2026ג, מועד 84, שאלה 9; 2026א, מועד 96, שאלות 7–9; 2026א, מועד 91, שאלות 6–7; 2026א, מועד 82, שאלות 9–10; 2025ג, מועד 91, שאלות 7, 14; 2025א, מועד 94, שאלה 6; 2025א, מועד 87, שאלה 6; 2024ג, מועד 84, שאלות 6, 11; 2024א, מועד 91, שאלה 5; 2023א, מועד 89, שאלה 7).

5 המעבד החד-מחזורי

5.1 אותות הבקרה ובקרת ה-ALU (D1)

ניסוח	כתיב
מעבד חד-מחזורי – כל פקודה מתבצעת במחזור שעון אחד ($CPI = 1$), ולכן מחזור השעון נקבע לפי הפקודה הארוכה ביותר – `lw` : זיכרון פקודות, קריאת אוגרים, ALU, זיכרון נתונים, כתיבה לאוגר. פקודה קצרה ממנה מחכה ב"זמן מת" עד סוף המחזור. (מדריך עמ' 131)	$CCT = T_{lw}, CPI = 1$

קווי אפשור (enable) – `RegWrite`, `MemWrite`, `MemRead` – חייבים להיות 0 או 1 בכל פקודה, כי הם משנים מצב. **בוררים** של מרבבים (`RegDst`, `ALUSrc`, `MemtoReg`) – (don't care) X כשהפקודה לא משתמשת בתוצאה, כלומר כש-`RegWrite = 0`. (עמ' 125, 128, 135)	`RegDst: 0 = rt, 1 = rd`
	`ALUSrc: 0 = Read data 2, 1 = SE(imm)`
	`MemtoReg: 0 = ALU, 1 = Mem`

הסתעפות וקפיצה – `PCSrc = Branch AND Zero` : ה-ALU מחסר ב-`beq` ו-`Zero = 1` כשהאוגרים שווים. `Jump = 1` בורר את כתובת הקפיצה ועוקף את `PCSrc`, ולכן רק ב-`j` כתוב X ב-`Branch`. (עמ' 123, 129–130, 135)	$PC = PC + 4 + SE(imm) \cdot 4$ אם $Branch \cdot Zero = 1$

טבלת הבקרה – כמו בתשובת המדריך (טבלה 4.18 בספר ועמודת Jump). השורה של `addi` אינה בטבלת המדריך ; היא נגזרת באותם כללים. (עמ' 135)

Instruction	Opcode	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0	Jump
R-type	0	1	0	0	1	0	0	0	1	0	0
lw	35	0	1	1	1	1	0	0	0	0	0
sw	43	X	1	X	0	0	1	0	0	0	0
beq	4	X	0	X	0	0	0	1	0	1	0
j	2	X	X	X	0	0	0	X	X	X	1
addi	8	0	1	0	1	0	0	0	0	0	0

בקרת ה-ALU – שני שלבים : הבקרה הראשית מפיקה `ALUOp` מה-opcode, ויחידת בקרת ה-ALU מצרפת אליו את 6 הסיביות `Instr[5-0]` (funct) ומפיקה 4 סיביות `ALUCtrl`. ב-`ALUOp = 00` מחברים (`lw`, `sw`), ב-`01` מחסרים (`beq`), וב-`10` הפעולה לפי funct (פקודות R). (עמ' 122, 127 ; טבלה 4.12 בספר)

ALUOp	funct	הפעולה	ALUCtrl
00	X	add	0010
01	X	sub	0110
10	0x20	add	0010
10	0x22	sub	0110
10	0x24	and	0000
10	0x25	or	0001
10	0x2a	slt	0111
10	0x27	nor	1100

תבנית – "מה ערכי הבקרה / מה נכנס ל-ALU / אות תקוע ב-0 או ב-1"

1. **פענוח:** מהקסדצימלי לשדות (4.6) : ה-opcode קובע את שורת הבקרה ; בפקודת R ה-funct קובע את הפעולה.
2. **שורת RTL** של הפקודה, למשל `$rt ← Mem[$rs + SE(imm)]` – ממנה רואים אילו יחידות כותבות ואילו מרבבים משנים.
3. **כל אות:** קו אפשור – 1 אם היחידה כותבת/קוראת, אחרת 0. בורר – הכניסה שהפקודה צריכה, ו-X אם התוצאה שלו לא נכתבת לשום מקום.
4. **ה-ALU:** כניסה עליונה תמיד `$rs = Read data 1` ; תחתונה לפי `ALUSrc`. `ALUCtrl` מ-`ALUOp` ומ-`Instr[5-0]`, גם בפקודת I (אז אלה 6 הסיביות הנמוכות של ה-immediate).
5. **כל קו בנתיב מחושב** גם אם התוצאה לא נבחרת : Read data 2, מחבר ההסתעפות, הרחבת הסימן.
6. **אות תקוע:** פקודה נפגעת אם בשורה שלה יש ערך (לא X) ששונה מהערך התקוע. בשאלת אחוזים – סוכמים את חלקי הפקודות שנפגעו.

דוגמה – המילה `0x8D2A0008` כש-`$t1` = `0x10010000` ובזיכרון `Mem[0x10010008]` = `0x2A` (datapath.py)

`lw $t2,8($t1)` = `0x8D2A0008` : op `35`, rs `9`, rt `10`, imm `8`. שורת `lw`, ובקרת ה-ALU : `ALUOp = 00` ⟵ `0010` (חיבור). RTL : `$t2 ← Mem[$t1 + SE(8)]`.

קו	ערך	למה
Read register 1 / 2	9 / 10	rs / rt
Write register	10	rt ⟵ `RegDst = 0`
כניסות ה-ALU	0x10010000, 0x00000008	SE(imm) ⟵ `ALUSrc = 1`
ALU result	0x10010008	הכתובת ; Zero = 0
Read data	0x0000002A	נכתב ל-`$t2` ⟵ `MemtoReg = 1`
PC+4	0x00400008	
מחבר ההסתעפות	0x00400028	מחושב, לא נבחר (`Branch = 0`)

אות תקוע – מי נפגע (הכלל של שלב 6 על הטבלה) : (עמ' 125, 135)

האות התקוע	נפגעות	לא נפגעות
`RegDst = 0`	R-type (כותבות ל-rt במקום ל-rd)	lw, sw, beq, j, addi
`ALUSrc = 1`	R-type, beq	lw, sw, j, addi
`MemtoReg = 1`	R-type, addi	lw, sw, beq, j
`RegWrite = 1`	sw, beq, j (דורסות אוגר)	R-type, lw, addi

שימו לב:

- **`Branch` הוא 0, לא X**, בכל פקודה שאינה הסתעפות – הוא נכנס לשער AND שקובע את ה-PC. רק ב-`j` כתוב X, כי `Jump` עוקף. ב-`j` : `RegWrite`, `MemRead`, `MemWrite` = 0 וכל השאר X (2021ג, מועד 83, שאלה 7).
- **funct של פקודת I:** `Instr[5-0]` מגיע לבקרת ה-ALU גם ב-`addi` ; ערך הקו הוא 6 הסיביות הנמוכות של ה-immediate. מי שכתב ערך אחר הפסיד את כל 3 הנקודות של הקו (2025א, מועד 94, שאלה 11.1).
- **`ALUOp` תקוע ב-10:** ב-`0xAC01FF22` = `sw $1,-222($0)` שדה ה-"funct" הוא `0x22` ⟵ `0110` (חיסור), וה-ALU מחשב $222 = (-222) - 0$. התשובה "לא ניתן לדעת" שגויה (2025א, מועד 94, שאלה 9).
- **Read data 2 נקרא תמיד**, אבל משמש רק כש-`ALUSrc = 0` או `MemWrite = 1` : R-type, beq, sw (2025א, מועד 81, שאלה 5 ; 2025ג, מועד 91, שאלה 9).

באילו מבחנים: (2026א, מועד 96, שאלה 9 ; 2026א, מועד 91, שאלה 9 ; 2026א, מועד 82, שאלה 9 ; 2025ג, מועד 91, שאלות 7, 9 ; 2025א, מועד 94, שאלה 9 ; 2025א, מועד 87, שאלה 7 ; 2025א, מועד 81, שאלה 5 ; 2024ג, מועד 84, שאלות 8, 11 ; 2024א, מועד 91, שאלה 10 ; 2023א, מועד 89, שאלה 10 ; 2022ג, מועד 92, שאלה 9 ; 2022א, מועד 82, שאלות 6, 8 ; 2021ג, מועד 83, שאלות 6, 7 ; 2021א, מועד 89, שאלות 3, 5).

5.2 פקודה או יחידה חדשה בנתיב (D2)

ניסוח	כתיב
מרבב עם כניסה 2 – כשמוסיפים פקודה, מוסיפים כניסה למרבב קיים ונותנים לבורר שלו ערך חדש (2). בדוגמת המדריך : כניסה 2 של `RegDst` = `Instr[31-27]`, כניסה 2 של `MemtoReg` = `rt`, ולכן opcode `0x33` כותב את `$rt` לאוגר `$t9` (11001). (עמ' 130–131, תשובה עמ' 135)	`0xCD2A0010` עם `$t2` = `0x1234` : `$t2 ← $t9` = `0x1234`
יחידה על קו (הזזה, NEG, NOT…) – משנה רק את מה שעובר בקו שעליו היא יושבת. כל הקווים שמתפצלים **לפני** הנקודה השחורה מקבלים את הערך המקורי. (עמ' 131 ; השאלות הפתוחות בכל מבחן)	קו ⟵ ● ⟵ יחידה ⟵ מרבב

תבנית – "הוסיפו פקודה" / "מה ערכי הקווים במעבד המשונה"

1. **שורת RTL** של מה שהפקודה צריכה לעשות (למשל `jal` : `$ra ← PC+4`, וה-PC לכתובת הקפיצה).
2. **טבלת מסלול:** לכל ערך – מאיזו יחידה הוא בא ובאיזו כניסה של איזה מרבב הוא נכנס. כניסה חדשה – מספר 2.
3. **שורת בקרה** עם הערכים החדשים (2 בבורר שהורחב), X בכל מה שלא משנה.

4. **שאלת ערכי קווים:** קודם **מצב המכונה** – כל פקודה ברשימה מפוענחת, וטבלת אוגרים וזיכרון לפני כל פקודה (מעקב, 4.4) עם השינוי שהיחידה גורמת. רק אז ערך לכל קו, ובשורה – מה מקורו (PC+4, imm, rd, תוצאת ALU, don't care).
5. **הקסדצימלי עם 0x**, ו-X רק לקו שערכו באמת לא ידוע.

דוגמה – הוספת `jal` (datapath.py) עם כניסה 2 ב-`RegDst` = 31 וב-`MemtoReg` = PC+4)

`jal 0x00400040` בכתובת `0x00400010` היא `0x0C100010`; נכתב `$ra` = `0x00400014`, וה-PC הבא `0x00400040`.

שלב	יחידה	כניסה
כתובת היעד	`RegDst`	כניסה 2 = 31
הערך הנכתב	`MemtoReg`	כניסה 2 = PC+4
ה-PC הבא	מרבב Jump	1

אות	ערך
RegDst / MemtoReg	2 / 2
RegWrite	1
MemRead / MemWrite / Branch	0 / 0 / 0
ALUSrc / ALUOp	X / XX
Jump	1

שימו לב:

- **קוד הוא גם נתונים:** כש-`lw` קורא מכתובת שבתוך רשימת הקוד, הערך הוא **קידוד הפקודה** שם, לא 0 ולא X (2025א, מועד 94, שאלה 11.1; `0xAC86FF80`; 2024ג, מועד 84, שאלה 15.1).
- **הקסדצימלי:** opcode שנכתב בבינרי עם `0x` לפניו (`0x101011`) עלה נקודה (2024ג, מועד 84, שאלה 15.2).
- **קו שלא נבחר עדיין מחושב:** פלט מחבר ההסתעפות נשאל גם כש-`PCSrc = 0` (2025א, מועד 94, שאלה 11.1).
- `andi` צריכה הרחבת **אפסים** – יחידה חדשה או בורר חדש על הרחבת הסימן, וקוד ALUOp שגורם ל-AND (2023א, מועד 89, שאלה 8).
- **אותות הבקרה באים מה-opcode:** גם כשה-funct "הופך" פקודה, הבקרה הראשית מפענחת את opcode 0 כ-R-type (2026ג, מועד 84, שאלה 13.3).

באילו מבחנים: החלק החד-מחזורי של השאלה הפתוחה הגדולה (30–37 נקודות) בכל מבחן: (2026ג, מועד 84, שאלה 13.1; 2026א, מועד 96, שאלה 13.1; 2026א, מועד 91, שאלה 13.1; 2026א, מועד 82, שאלה 13.1; 2025ג, מועד 91, שאלה 11.1; 2025א, מועד 94, שאלה 11.1; 2025א, מועד 87, שאלה 11.1; 2025א, מועד 81, שאלה 11.1; 2024ג, מועד 84, שאלה 15.1; 2024א, מועד 91, שאלה 15.1; 2023א, מועד 89, שאלות 8, 15.1; 2023א, מועד 67, שאלה 15.1; 2022ג, מועד 92, שאלה 15.1; 2022ג, מועד 65, שאלה 15.1; 2022א, מועד 82, שאלה 12.1; 2022א, מועד 75, שאלה 12.1; 2021ג, מועד 83, שאלה 14א; 2021א, מועד 89, שאלה 8.1; 2021א, מועד 83, שאלה 8.1).

5.3 מחזור השעון והמסלול הקריטי (D3)

ניסוח	כתיב
זמן של מחלקת פקודות – סכום זמני היחידות שהיא עוברת בהן, לפי הסדר. המחזור = הארוך ביותר. (עמ' 131, 138; תרשים 4.26 בספר)	R: IF + Reg + ALU + Reg lw: IF + Reg + ALU + Mem + Reg sw: IF + Reg + ALU + Mem beq: IF + Reg + ALU ·j: IF
זמני הגעה – כשנתונים זמנים של כל יחידה (גם מרבבים), זמן ההגעה ליחידה = המקסימום על הכניסות שלה ועוד ההשהיה שלה. מרבב שהפקודה לא משתמשת בו (X) לא על המסלול. (עמ' 131)	$t(u) = \max_{in} t(in) + d(u)$

תבנית – "מה מחזור השעון / מה המסלול הקריטי / כמה מהר אחרי שינוי יחידה"

1. **טבלת מחלקות:** מחלקה | היחידות | סכום – לכל מחלקה, גם אם שואלים רק על אחת.
2. **מחזור השעון** = הסכום הגדול ביותר, ומי שקבעה אותו.
3. **עם השהיות של מרבבים:** טבלת יחידה | השהיה | זמן הגעה מצטבר לאורך המסלול; בכל צומת – הכניסה המאוחרת.
4. **שינוי יחידה:** מחשבים מחדש **את כל המחלקות** – המחלקה הארוכה יכולה להתחלף. היחס = מחזור ישן חלקי חדש (ה-CPI נשאר 1).

דוגמה – IMem 200, Reg 100, ALU 200, DMem 200, כתיבה לאוגר 100 ps (datapath.py)

מחלקות: R 600, lw **800**, sw 700, beq 500, j 200 – המחזור 800 ps, נקבע על ידי `lw`. עם 25 ps לכל מרבב, המסלול של `lw`:

unit	PC	IMem	RegRead	ALU	DMem	MemtoReg mux	RegWrite
delay	0	200	100	200	200	25	100
arrival	0	200	300	500	700	725	**825**

מרבב ה-ALUSrc מוכן ב-225, לפני Read data 1 (300), ולכן אינו על המסלול של lw. ב-R-type הוא כן: 650.

שימו לב:

- **חד-מחזורי מול צנרת:** בחד-מחזורי $CPI = 1$ והמחזור הוא הפקודה הארוכה; צנרת לא מקצרת פקודה בודדת – היא מאריכה אותה ומשפרת תפוקה (2025א, מועד 94, שאלה 7; 2021ג, מועד 83, שאלה 8).

באילו מבחנים: (2025א, מועד 94, שאלה 7; 2023א, מועד 67, שאלה 10; 2022ג, מועד 65, שאלה 8; 2021ג, מועד 83, שאלה 8; 2021א, מועד 83, שאלה 5) – לרוב יחד עם מחזור השעון של הצנרת (6.1).

6 הצנרת

6.1 דיאגרמת הצנרת, מחזורים ו-CPI (F1)

ניסוח	כתיב
חמשת השלבים – IF (הבאת פקודה), ID (פענוח וקריאת אוגרים), EX (ALU), MEM (זיכרון נתונים), WB (כתיבה לאוגר). בין כל שני שלבים אוגר צנרת: IF/ID, ID/EX, EX/MEM, MEM/WB. מחזור השעון = **השלב הארוך ביותר**. (מדריך עמ׳ 138, 144)	$CCT_{pipe} = \max(T_{IF}, T_{ID}, T_{EX}, T_{MEM}, T_{WB})$
זמן ותפוקה – n פקודות בלי סיכונים ב-k שלבים: $k+n-1$ מחזורים, ו-CPI שואף ל-1. פקודה בודדת לא מתקצרת – היא לוקחת k מחזורים ארוכים. (עמ׳ 138)	$Speedup = \frac{T_{single} \cdot 1}{CCT_{pipe} \cdot CPI_{pipe}}$
חציית מקבץ האוגרים – כתיבה בחצי הראשון של המחזור וקריאה בחצי השני, לכן פקודה שקוראת אוגר 3 פקודות אחרי הכותבת (WB ו-ID באותו מחזור) מקבלת את הערך החדש. (עמ׳ 145)	WB של (1) ו-ID של (4) – באותו מחזור

תבנית – ״כמה מחזורים / מה ה-CPI / באיזה מחזור קורה X״

1. **הגריד:** שורה לכל פקודה, עמודה לכל מחזור CC1, CC2…, ובתאים IF ID EX MEM WB. כל פקודה מתחילה מחזור אחד אחרי הקודמת.
2. **השהיה:** הפקודה התקועה **חוזרת על השלב** שלה (ID ID), וגם זו שאחריה (IF IF); מתחתיה שורת ״bubble״ שממשיכה EX MEM WB.
3. **ריקון** (flush): התא של הפקודה שבוטלה מסומן בקו חוצה, והיא לא ממשיכה.
4. **סה״כ מחזורים** = המחזור של ה-WB האחרון; $CPI = \frac{\text{מחזורים}}{IC}$ (או $IC/(4-\text{מחזורים})$ במצב יציב, אם כך מוגדר).
5. **שאלת האצה:** מחזור חד-מחזורי = סכום השלבים, מחזור צנרת = הגדול, ומכפילים כל אחד ב-CPI שלו.

דוגמה – עם העברה קדימה (pipeline.py), `$s0 = 0x100, Mem[0x100] = 2, $s1 = 5, $t0 = 10`

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9
`lw $t0,0($s0)`	IF	ID	EX	MEM	**WB**				
`add $t1,$t0,$s1`		IF	ID	ID	**EX**	MEM	WB		
bubble				EX	MEM	WB			
`sub $t2,$t1,$s2`			IF	IF	ID	EX	MEM	WB	
`sw $t2,4($s0)`					IF	ID	EX	MEM	WB

9 מחזורים, $IC = 4$, $CPI = 9/4$; השהיה אחת (load-use) ב-CC4. בסוף: `$t0 = 10`, `$t1 = 15`, `$t2 = 13`.

האצה (תשובת המדריך): שלבים של 300, 400, 350, 500, 100 ps – חד-מחזורי 1,650 ps, צנרת 500 ps, האצה אידיאלית $1650/500 = 3.3$; אבל `lw` לבד לוקחת בצנרת $5 \cdot 500 = 2500$ ps. (עמ׳ 163)

שימו לב:

- **CPI של הצנרת:** כשנתון CPI של הצנרת, ההאצה היא $\frac{800 \cdot 1}{200 \cdot 2} = 2$ (2023א, מועד 67, שאלה 10) ו-$\frac{1600 \cdot 1}{400 \cdot 4} = 1$ (2022ג, מועד 65, שאלה 8) – לא יחס המחזורים לבד.
- **פיצול השלב הארוך:** המחזור יורד רק עד השלב הארוך **הבא** (2021א, מועד 83, שאלה 5).
- **בלי גריד מפסידים:** תשובות על שאלות הצנרת בלי מפת שלבים הפסידו עד 7 נקודות. מציירים אותו בכל שאלת צנרת, גם כשלא ביקשו.

באילו מבחנים: (2024ג, מועד 84, שאלה 10; 2023א, מועד 67, שאלה 10; 2022ג, מועד 65, שאלה 8; 2021א, מועד 83, שאלה 5), והגריד הוא הצעד הראשון בכל שאלות 6.2–6.6.

6.2 סיכוני נתונים: העברה קדימה והשהיה (F2)

ניסוח	כתיב
סיכון נתונים – פקודה קוראת אוגר לפני שהפקודה שכותבת אותו הגיעה ל-WB. **העברה קדימה** (forwarding) לוקחת את הערך מאוגר הצנרת ישר לכניסת ה-ALU: מ-EX/MEM (הפקודה הקודמת) או מ-MEM/WB (שתיים לפני). (עמ' 145–147)	ForwardA / ForwardB: 00 = מקבץ האוגרים (כניסה 0) 10 = EX/MEM (כניסה 2) 01 = MEM/WB (כניסה 1)

תנאי יחידת ההעברה – בכל אחד גם `RegWrite = 1` אצל הכותבת ו-`RegisterRd` $\neq 0$. כשגם 1 וגם 2 מתקיימים – EX/MEM קודם (הערך החדש יותר). (עמ' 146, 149; הספר עמ' 318–323)

מקרה	התנאי	הבורר	הכניסה
1a	`EX/MEM.RegisterRd = ID/EX.RegisterRs`	ForwardA = 10	2
1b	`EX/MEM.RegisterRd = ID/EX.RegisterRt`	ForwardB = 10	2
2a	`MEM/WB.RegisterRd = ID/EX.RegisterRs` (ואין 1a)	ForwardA = 01	1
2b	`MEM/WB.RegisterRd = ID/EX.RegisterRt` (ואין 1b)	ForwardB = 01	1

ניסוח	כתיב

load-use – הנתון של `lw` מוכן רק בסוף MEM, ולכן הפקודה הבאה שקוראת אותו חייבת לחכות מחזור. יחידת איתור הסיכונים (HDU) ב-ID: אם `ID/EX.MemRead = 1` ו-`ID/EX.RegisterRt` שווה ל-`IF/ID.RegisterRs` או ל-`IF/ID.RegisterRt` – משהים: `PCWrite = 0`, `IF/IDWrite = 0`, ומרבב הבקרה מכניס אפסים ל-ID/EX (בועה). (עמ' 148–149; הספר עמ' 326)	`PCWrite = 0`
	`IF/IDWrite = 0`
	control → 0

load-store – `lw` ואחריה `sw` של אותו rt: אפשר להעביר מ-MEM/WB לכניסת ה-data של הזיכרון בלי השהיה; ה-HDU של הספר בכל זאת משהה מחזור. (עמ' 149)	`lw $9,100($10)` `sw $9,200($11)`

כמה השהיות (pipeline.py, `beq` מוכרעת ב-ID):

התלות	עם העברה קדימה	בלי העברה (רק חציית מקבץ האוגרים)
פקודת ALU → הפקודה הבאה	0	2
פקודת ALU → מרחק 2	0	1
פקודת ALU → מרחק 3	0	0
`lw` → הפקודה הבאה (load-use)	1	2
`lw` → מרחק 2	0	1
`lw` → `sw` של אותו rt	1 (0 עם העברה MEM/WB → זיכרון)	2
פקודת ALU → `beq` ב-ID	1	2
`lw` → `beq` ב-ID	2	2

תבנית – "מצאו את סיכוני הנתונים ואיך כל אחד נפתר"

1. **רשימת תלויות:** לכל פקודה – אילו אוגרים היא קוראת (rs, ו-rt רק אם היא משתמשת בו), ומי מ-3 הפקודות הקודמות כותבת אותם. `beq` ו-`sw` לא כותבות.
2. **הגריד** (6.1) עם ההשהיות לפי הטבלה למעלה.
3. **טבלת סיכונים:** אוגר | בין פקודות | מקרה (load-use, 1a/1b/2a/2b, חציית מקבץ) | פתרון | הבורר והמחזור.
4. **תלויות שאינן סיכון:** אומרים במפורש (מרחק 3, פקודה שלא כותבת, בועה שהרחיקה).
5. **קווי ה-HDU** במחזור ההשהיה: `PCWrite`, `IF/IDWrite`, בורר מרבב הבקרה.

דוגמה – הקוד של 6.1 (pipeline.py)

אוגר	בין פקודות	מקרה	פתרון	Forward
$t0	(1) → (2)	load-use, ואז 2a	השהיה אחת + העברה מ-MEM/WB	ForwardA = 01 (כניסה 1), CC5
$t1	(2) → (3)	1a	העברה מ-EX/MEM	ForwardA = 10 (כניסה 2), CC6
$t2	(3) → (4)	1b	העברה מ-EX/MEM	ForwardB = 10 (כניסה 2), CC7

שימו לב:

- **Rd** $\neq 0$**:** בלי הבדיקה, `addi $0,$1,5` (עם `$1` = 7) "מעבירה" 12 ל-`$0`, `add $2,$0,$0` ו-`$2` מקבל 24 במקום 0 (2023א, מועד 89, שאלה 11).
- **EX/MEM לפני MEM/WB:** בשלוש `add` רצופות על `$1`, השלישית לוקחת מ-EX/MEM (מקרה 1a), לא מ-MEM/WB. (תשובה 9, עמ' 164)
- **ה-HDU הפורמלי** משווה גם ל-`IF/ID.RegisterRt` כשהפקודה לא קוראת את rt (`addi`), ולכן משהה בלי צורך; המדריך מעיר שהתנאי חלקי. עונים לפי מה שהשאלה מתארת ומסבירים (2023א, מועד 67, שאלה 15.2). (הערה 1, עמ' 148)
- **יחידת ההעברה לא בודקת שימוש:** ForwardB = 10 גם ל-`lw` שלא קוראת rt – וזה קיבל ניקוד מלא (2025א, מועד 94, שאלה 11.2).
- **בועה חלקית:** אם הבועה הייתה משאירה `MemRead = 1`, ה-HDU היה משהה בלי סוף – לכן מאפסים את **כל** קווי הבקרה (2023א, מועד 67, שאלה 9).

באילו מבחנים: (2025א, מועד 81, שאלה 10; 2023א, מועד 89, שאלה 11; 2023א, מועד 67, שאלה 9; 2022ג, מועד 92, שאלה 10; 2022ג, מועד 65, שאלה 10; 2021א, מועד 89, שאלה 8.2), ובכל חלק הצנרת של השאלה הפתוחה (6.5).

6.3 סיכוני בקרה (F3)

ניסוח	כתיב
הכרעה ב-ID – משווה במקבץ האוגרים ומחבר היעד עוברים ל-ID, ולכן כש-beq נלקחת רק הפקודה שב-IF מתבטלת (`IF.Flush` הופך אותה ל-nop); מחיר מחזור אחד. כשההכרעה ב-MEM (תרשים 4.51 בספר) – שלוש פקודות. (עמ' 152–154)	נלקחת: ID − 1, EX − 2, MEM − 3 לא נלקחת (חיזוי "לא נלקחת"): 0
המשווה ב-ID צריך את הערכים כבר ב-ID – אחרי פקודת ALU שכותבת את האוגר: השהיה אחת; אחרי `lw`: שתיים. `j` מוכרעת ב-ID ומבטלת פקודה אחת. (עמ' 153; תשובה 12ג, עמ' 166)	$CPI = 1 + f_{beq} \cdot p_{taken} \cdot penalty$
קפיצה מושהית (delayed branch) – הפקודה שאחרי ה-beq מתבצעת **תמיד**, והמהדר שם בה פקודה מועילה. **חיזוי דינמי** – 1 סיבית (משנה כיוון אחרי טעות אחת) או 2 סיביות (רק אחרי שתי טעויות); BTB שומר את כתובת היעד כבר ב-IF. (עמ' 154–157)	2 סיביות: ST, WT (חוזה "נלקחת"), WNT, SNT (חוזה "לא נלקחת")

תבנית – "כמה עולה הסתעפות / מה ה-CPI / מה החיזוי"

1. **איפה מוכרעת** ההסתעפות (ID, EX, MEM), ולכן כמה פקודות מתבטלות כשהיא נלקחת.
2. **הגריד** עם התא המבוטל מסומן; מתחתיו שורת מחיר לכל הסתעפות: "הוכרעה ב-X, אובדים N מחזורים", ועוד ההשהיות של המשווה ב-ID.
3. **CPI:** $(\text{מחיר}) \cdot (\text{חלק הנלקחות}) \cdot (\text{חלק ההסתעפות}) + 1$, בשברים מדויקים.
4. **חיזוי:** טבלה עם עמודה לכל ביצוע – מצב לפני, חיזוי, בפועל, נכון/לא, מצב אחרי. **מצב ההתחלה** כתוב בשאלה – בודקים אותו בציור.

דוגמה – `beq $s1,$s2,L` נלקחת (pipeline.py, `$s1` = `$s2`)

	CC1	CC2	CC3	CC4	CC5	CC6	CC7
`beq $s1,$s2,L`	IF	**ID**	EX	MEM	WB		
`add $t0,$s1,$s2`		~~IF~~					
`L: sub $t2,$s1,$s2`			IF	ID	EX	MEM	WB

- הוכרעה ב-ID ב-CC2, פקודה אחת בוטלה: מחזור אחד. אותו קוד עם הכרעה ב-MEM מבטל 3 פקודות.
- 25% הסתעפויות, 40% מהן נלקחות: הכרעה ב-ID – $CPI = 1 + 0.25 \cdot 0.4 \cdot 1 = 1.1$; ב-MEM – $1 + 0.25 \cdot 0.4 \cdot 3 = 1.3$.
- לולאה של 4 סיבובים שרצה פעמיים: T T T NT T T T NT. חיזוי 2 סיביות מ-ST: נכון 6 מתוך 8 (טועה רק ביציאה); 1 סיבית: 5 מתוך 8 (טועה גם בכניסה החוזרת).

שימו לב:

- **IF.Flush מול IFWrite = 0:** הראשון מאפס את IF/ID (הפקודה הופכת ל-nop), השני משאיר את התוכן כמו שהוא (השהיה); הבועה עצמה באה ממרבב הבקרה (2025א, מועד 81, שאלה 10).

- **בלי `IF.Flush`** הפקודה שאחרי beq נלקחת מתבצעת – כמו קפיצה מושהית (2024ג, מועד 84, שאלה 9).
- **הכרעה ב-MEM** מבטלת $0.2 \cdot 0.2 \cdot 3 = 0.12$ מחזורים לפקודה (2022א, מועד 82, שאלה 5).
- **מצב ההתחלה של החיזוי:** בשאלה 13 של המדריך התשובה 40% מתקבלת רק מהמצב החזק "לא נלקחת"; מהמצב החלש יוצא 0%. כותבים מאיזה מצב התחלתם. (עמ' 155–156, תשובה עמ' 166)

באילו מבחנים: (2026א, מועד 91, שאלה 6; 2026א, מועד 82, שאלה 10; 2025א, מועד 87, שאלות 4, 10; 2025א, מועד 81, שאלה 10; 2024ג, מועד 84, שאלה 9; 2024א, מועד 91, שאלה 14; 2023א, מועד 67, שאלה 12; 2022ג, מועד 92, שאלות 8, 10; 2022א, מועד 82, שאלה 5; 2021ג, מועד 83, שאלות 9, 10).

6.4 אוגרי הצנרת במחזור N (F4)

ניסוח	כתיב
מה עובר בכל אוגר – IF/ID: PC+4 והפקודה. ID/EX: הבקרה, PC+4, Read data 1/2, SE(imm), rs, rt, rd. EX/MEM: הבקרה של WB ו-M, תוצאת ה-ALU, Zero, Read data 2 (הנתון ל-sw), ה-Rd שנבחר. MEM/WB: הבקרה של WB, הנתון מהזיכרון, תוצאת ה-ALU, Rd. (עמ' 141–145)	`MemtoReg`, `RegWrite`: WB `MemWrite`, `MemRead`, `Branch`: M `ALUOp`, `ALUSrc`, `RegDst`: EX
הבקרה נוצרת ב-ID ועוברת עם הפקודה; כל שלב משתמש בקבוצה שלו ומעביר את השאר. ב-EX/MEM כבר אין `RegDst`, `ALUSrc`, `ALUOp`. (עמ' 144–145; טבלה 4.49, תרשים 4.51)	ID/EX: WB + M + EX EX/MEM: WB + M MEM/WB: WB
כלל הבועה – ה-HDU מאפס **רק את קווי הבקרה** שנכנסים ל-ID/EX. שאר השדות (rs, rt, rd, הערכים שנקראו, ה-immediate) הם של הפקודה שהושהתה, ויחידת ההעברה ממשיכה לפעול עליהם. (עמ' 149)	בועה = הפקודה המושהית עם בקרה 0

תבנית – "מה הערך על קווים N–1 במחזור N"

1. **מפת שלבים** של מחזור N מהגריד: WB | MEM | EX | ID | IF – איזו פקודה (או בועה) בכל שלב.
2. **מצב המכונה:** ערכי האוגרים והזיכרון שהפקודות האלה קוראות (מעקב), ואיזו העברה קדימה פעילה (6.2).
3. **לכל קו:** שם השדה (`EX/MEM.RegisterRd`, `ForwardA`, `ALU.result`...), מאיזו פקודה הוא, והערך בהקסדצימלי.
4. **בועה:** בקרה 0, השדות של הפקודה המושהית; `RegDst = 0` מעביר את rt שלה, `ALUOp = 00` מחבר את הערכים שלה.

דוגמה – מחזור 6 בקוד של 6.1 (pipeline.py)

IF	ID	EX	MEM	WB
(אין)	`sw $t2,4($s0)`	`sub $t2,$t1,$s2`	`add $t1,$t0,$s1`	bubble

שדה	ערך	מקור
ID/EX.RegisterRs	9	rs של sub ($t1)
ForwardA	10	EX/MEM ב-$t1 (מקרה 1a)
ALU.A	15	EX/MEM.ALUResult של add
ALU.result	13	15 − 2
EX/MEM.RegisterRd	9	rd של add
MEM/WB.RegWrite	0	בועה

הבועה (כלל עמ' 149)	ערך	למה
CC4 ב-rs / rt / rd	8 / 17 / 9	של add, כפי שנקראו ב-CC3
CC4 ב-ForwardA	10	8 = EX/MEM.RegisterRd rs = (1w)
CC4 ב-ALU.result	0x105	5 + 0x100 :ALUOp = 00
CC4 ב-RegDst.out	17	rt → RegDst = 0
CC6 ב-MEM/WB.RegisterRd	17	לא X ולא 0
CC6 ב-MEM/WB.ALUResult	0x105	

שימו לב:

- **הבועה שומרת את השדות:** המפתח נתן 0xB (rt של הפקודה המושהית) לקו היעד של הבועה (2021א, מועד 89, שאלה 8.2), ו-0x05 לפלט RegDst של בועה – rt שלה אחרי היחידה (2022ג, מועד 65, שאלה 15.2). בועה ב-EX עדיין מקבלת העברה קדימה ומחברת את הערכים של הפקודה המושהית (2026א, מועד 82, שאלה 13.2; 2026א, מועד 96, שאלה 13.2).
- **כניסה 1 של מרבב ההעברה** היא הערך שנכתב ב-WB (פלט `MemtoReg`), לא הערך שבמקבץ האוגרים (2026א, מועד 91, שאלה 13.2).

- **ב-EX/MEM אין RegDst** – רק Branch, MemRead, MemWrite, MemtoReg, RegWrite (2025ג, מועד 91, שאלה 10). הבקרה עוברת דרך אוגרי הצנרת, לא ישר מיחידת הבקרה (2026ג, מועד 84, שאלה 8).
- **ערך לכל קו בהקסדצימלי עם 0x**; X עם סיבה קצרה ("RegDst הוא don't care ב-sw") קיבל ניקוד מלא (2022א, מועד 82, שאלה 12.2).

באילו מבחנים: (2026ג, מועד 84, שאלה 8 ; 2025ג, מועד 91, שאלה 10) ובכל חלק הצנרת של השאלה הפתוחה – הרשימה ב-6.5.

6.5 יחידה או פקודה חדשה בצנרת (F5)

ניסוח	כתיב
איפה היחידה יושבת קובע הכול: לפני ID/EX – משנה את מה שנקרא ממקבץ האוגרים, אבל ערך שמגיע בהעברה קדימה **עוקף** אותה. על EX/MEM אחרי הפיצול – הזיכרון ויחידת ההעברה מקבלים את הערך המקורי, ורק MEM/WB את המשונה. (עמ' 141–149 ; השאלות הפתוחות)	נקודה שחורה לפני היחידה = הפיצול לוקח את הערך **המקורי**
מספרי האוגרים – ה-HDU משווה את ID/EX.RegisterRt כפי שהוא נכנס אליו, ויחידת ההעברה את EX/MEM.RegisterRd, MEM/WB.RegisterRd כפי שהם יוצאים. יחידה על אחד הקווים האלה משנה את ההחלטה רק בקו שלה. (עמ' 146–149)	HDU: rt לפני היחידה Forwarding unit: Rd אחרי היחידה

תבנית – "יחידה חדשה בצנרת: ערכי הקווים במחזור N"

1. **מיקום:** באיזה שלב, לפני או אחרי אוגר הצנרת, לפני או אחרי הפיצול להעברה קדימה, ל-HDU ולזיכרון.
2. **הגריד ומפת השלבים** של מחזור N (6.1, 6.4) – עם ההשהיות כפי שה-HDU **רואה** את המספרים.
3. **טבלת שלבים לכל פקודה:** שלב | מה היא עושה | מה עובר לאוגר הבא (שדות ובקרה) – והיחידה מופעלת רק על מה שעובר דרכה.
4. **העברה קדימה:** מאיזה צד של היחידה יוצא הקו (מקורי או משונה), ומה זה עושה לכתובות ולערכים.
5. **השוואה לחד-מחזורי:** אם התוצאה שונה, זה בגלל ההעברה קדימה – כותבים את זה במשפט.

דוגמה – מה `lw $t0,0($s0)` נושאת מ-CC1 עד CC5 (pipeline.py, datapath.py)

שלב	העבודה	מה עובר הלאה
IF	IMem[PC], PC+4 = 4	PC+4, הפקודה (IF/ID)
ID	קורא $s0 = 0x100, SE(0)	WB{RegWrite=1, MemtoReg=1}, M{MemRead=1}, EX{ALUSrc=1, ALUOp=00, RegDst=0}, rt = 8 (ID/EX)
EX	0 + 0x100	ALUResult = 0x100, WB, M, Rd = 8 (EX/MEM)
MEM	Mem[0x100] = 10	ReadData = 10, WB, Rd = 8 (MEM/WB)
WB	10 ← $t0	–

שימו לב:

- **NEG על EX/MEM אחרי הפיצול:** sw קיבלה בהעברה את הערך **הלא משונה** לבסיס, וכתבה ל-0x3500 ולא ל-0x4B00 כמו בחד-מחזורי (2025א, מועד 87, שאלה 11.2).
- **הזזה אחרי נקודת ההעברה:** הערך שמועבר לא מוזז – 0x540 ולא 0x2C00 כמו במפתח הרשמי (2022א, מועד 75, שאלה 12.2).
- **יחידה לפני ID/EX** (NOT, NEG): ערך שמגיע בהעברה קדימה עוקף אותה (2026א, מועד 96, שאלה 13.2 ; 2024ג, מועד 84, שאלה 15.2).
- **ה-HDU מול יחידת ההעברה:** ה-HDU משהה לפי rt **לפני** היחידה (2023א, מועד 67, שאלה 15.2 ; 2022ג, מועד 65, שאלה 15.2), גם כשבגלל היחידה lw כותבת בכלל לאוגר אחר ואין ממה להעביר (2026א, מועד 82, שאלה 13.2).

באילו מבחנים: החלק של הצנרת בשאלה הפתוחה הגדולה (12–24 נקודות) בכל מבחן: (2026ג, מועד 84, שאלה 13.2 ; 2026א, מועד 96, שאלה 13.2 ; 2026א, מועד 91, שאלה 13.2 ; 2026א, מועד 82, שאלה 13.2 ; 2025ג, מועד 91, שאלה 11.2 ; 2025א, מועד 94, שאלה 11.2 ; 2025א, מועד 87, שאלה 11.2 ; 2025א, מועד 81, שאלה 11.2 ; 2024ג, מועד 84, שאלה 15.2 ; 2024א, מועד 91, שאלה 15.2 ; 2023א, מועד 89, שאלה 15.2 ; 2023א, מועד 67, שאלה 15.2 ; 2022ג, מועד 92, שאלה 15.2 ; 2022ג, מועד 65, שאלה 15.2 ; 2022א, מועד 82, שאלה 12.2 ; 2022א, מועד 75, שאלה 12.2 ; 2021ג, מועד 83, שאלה 14ב ; 2021א, מועד 89, שאלה 8.2 ; 2021א, מועד 83, שאלה 8.2).

6.6 חריגות (F6)

ניסוח	כתיב
שני סוגים בקורס – opcode לא מוגדר (מתגלה ב-ID, כשהבקרה רואה את ה-opcode) וגלישה ב-`add`, `sub`, `addi` (מתגלה ב-EX; `addu`, `addiu` לא גולשות). (עמ' 82–83, 157–158)	גלישה – בשלב EX opcode לא מוגדר – בשלב ID
EPC ו-Cause – EPC שומר את הכתובת של הפקודה שגרמה לחריגה **ועוד 4** (מה שהגיע מה-PC), והמטפל מחסיר 4. Cause אומר מה קרה (בספר: `12` גלישה, `10` opcode לא מוגדר). ה-PC עובר ל-`0x80000180`, כתובת המטפל של מערכת ההפעלה. לא `$ra`. (עמ' 158–159)	$EPC = PC_{fault} + 4$ $PC \leftarrow$ `0x80000180`
ריקון – הפקודה הגולשת והשתיים שאחריה מתבטלות: `EX.Flush`, `ID.Flush`, `IF.Flush`. היא **לא כותבת** את אוגר היעד; הפקודות שלפניה מסתיימות. המטפל נכנס ל-IF במחזור הבא. (עמ' 158–159, תרשימים 4.66–4.67)	`IF.Flush = ID.Flush = EX.Flush = 1`

תבנית – "מה קורה בחריגה / מה ב-EPC / אילו פקודות מסתיימות"

1. **איזו חריגה ובאיזה שלב** היא מתגלה: גלישה – EX (ערכי האוגרים בדיוק כמו בהעברה קדימה); opcode – ID.
2. **הגריד:** הפקודה הגורמת מסומנת בשלב שבו התגלתה, וכל הצעירות ממנה – בשלבים שלהן; המטפל ב-IF במחזור הבא.
3. **שורה לכל פלט:** EPC = הכתובת + 4, Cause, ה-PC החדש `0x80000180`, אילו אוגרים **לא** נכתבו, אילו פקודות הגיעו ל-WB.

דוגמה – גלישה ב-`add` (`$s1 = 0x7fffffff`, `$s2 = 1`, pipeline.py, הקוד מ-`0x40`)

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9
`0x40 sub $t1,$s3,$s4`	IF	ID	EX	MEM	WB				
`0x44 add $t0,$s1,$s2`		IF	ID	~~EX~~					
`0x48 or $t2,$s3,$s4`			IF	~~ID~~					
`0x4C lw $t3,0($s0)`				~~IF~~					
המטפל `0x80000180`					IF	ID	EX	MEM	WB

הגלישה מתגלה ב-CC4: EPC = `0x48`, Cause = `12`, `$t0` לא נכתב, ו-`sub` מסתיימת (`$t1` = 4).

שימו לב:

- **EPC ולא `$ra`:** `$ra` שייך לתוכנית, וחריגה יכולה לקרות בכל פקודה (2022א, מועד 75, שאלה 8).
- **opcode לא מוגדר** מתגלה כבר ב-ID: מתבטלות היא (ID.Flush) והבאה (IF.Flush), וה-PC הבא `0x80000180` (2022ג, מועד 92, שאלה 8; 2026א, מועד 82, שאלה 13.2).
- **`IF.Flush` פועל גם בחריגה**, לא רק בהסתעפות (2024ג, מועד 84, שאלה 9).
- **הקו שנכנס ל-EPC** נושא PC+4 – בודקים בציור מאיזה אוגר צנרת הוא יוצא (2024א, מועד 91, שאלה 15.2; 2026א, מועד 91, שאלה 13.2).

באילו מבחנים: (2026א, מועד 91, שאלה 13.2; 2026א, מועד 82, שאלה 13.2; 2024ג, מועד 84, שאלה 9; 2024א, מועד 91, שאלה 15.2; 2022ג, מועד 92, שאלה 8; 2022ג, מועד 65, שאלה 9; 2022א, מועד 75, שאלה 8).

7 זיכרון מטמון

7.1 חלוקת הכתובת (G1)

ניסוח	כתיב
בלוק, שורה, קבוצה – המטמון מחולק לבלוקים (שורות) של 2^m מילים. במיפוי ישיר כל בלוק של הזיכרון הולך לשורה אחת: $index = block \bmod \#lines$. ב-K-way יש $\#sets = \#blocks/K$ קבוצות, והבלוק יכול לשבת בכל אחת מ-K השורות של הקבוצה; באסוציאטיבי מלא – אין אינדקס. (מדריך עמ׳ 172–177, 182–184)	$block = \lfloor addr/\text{bytes per block} \rfloor$ $tag = \lfloor block/\#sets \rfloor \text{, } set = block \bmod \#sets$
רוחב השדות – n סיביות אינדקס (2^n קבוצות), m סיביות היסט מילה בבלוק, 2 סיביות היסט בית. התגית היא כל השאר. כתובת של 32 סיביות – אלא אם השאלה נותנת זיכרון אחר (2^{30} בתים = 30 סיביות). (עמ׳ 175–177)	$tag = 32 - (n + m + 2)$

תבנית – ״כמה סיביות בכל שדה / לאיזו קבוצה הולכת הכתובת / מה טווח הבלוק״

1. **מספרים בחזקות של 2:** גודל המטמון (נתונים בלבד), גודל הבלוק, K, ורוחב הכתובת.
2. **השדות:** byte offset = 2; $m = \log_2(\text{מילים בבלוק})$; $n = \log_2(\#blocks/K)$; tag = השאר.
3. **הכתובת בבינרי** מתחת לשדות, עם קו בכל גבול; כל שדה נקרא בנפרד (התגית מקובצת להקסדצימלי **מהסיבית הנמוכה שלה**).
4. **טווח הבלוק:** אותה כתובת עם כל סיביות ההיסט 0 ועד כולן 1.
5. **שינוי K:** כל הכפלה של K מורידה סיבית מהאינדקס ומוסיפה אחת לתגית.

דוגמה – מטמון 4KB, בלוק 16 בתים, הכתובת 0x1234ABCD (cache.py)

Name of the field	Tag	Index	Block offset	Byte offset
Number of bits	20	8	2	2
0x1234ABCD	00010010001101001010	10111100	11	01

האינדקס 188, התגית 0x1234A, והבלוק הוא הבתים 0x1234ABC0–0x1234ABCF. ב-4-way: 22 | 6 | 2 | 2, הקבוצה 60 – התגית גדלה ב-2. במדריך: הבית 1200 במטמון של 64 בלוקים של 16 בתים – בלוק 75, אינדקס 11, תגית 1.

שימו לב:

- **byte offset תמיד 2 סיביות**, ובכתובת 0x1765432F ערכו 3 (2022א, מועד 82, שאלה 10).
- **הקבוצה ב-K-way** היא שדה האינדקס בכתובת – $block \bmod \#sets$, ופחות קבוצות כש-K גדל. כל קבוצה שגויה עלתה נקודה (2025א, מועד 94, שאלה 14.2).
- **רוחב כתובת אחר:** 30 סיביות (2025א, מועד 81, שאלה 14), 28 (2025א, מועד 87, שאלה 14), 31 – ״2GB״ (2023א, מועד 67, שאלה 13), 36 (2021ג, מועד 83, שאלה 11).
- **״16 Kb״** – קילו**ביט**; אבל התגית גדלה ב-2 בין מיפוי ישיר ל-4-way בכל גודל (2026א, מועד 96, שאלה 11).

באילו מבחנים: כמעט בכל מבחן: (2026ג, מועד 84, שאלה 12; 2026א, מועד 96, שאלה 11; 2026א, מועד 91, שאלות 10, 11; 2026א, מועד 82, שאלה 12; 2025ג, מועד 91, שאלה 13; 2025א, מועד 94, שאלה 14; 2025א, מועד 87, שאלה 14; 2025א, מועד 81, שאלה 14; 2023א, מועד 67, שאלות 13, 14; 2022ג, מועד 92, שאלות 12, 13; 2022ג, מועד 65, שאלות 12, 13; 2022א, מועד 82, שאלות 9, 10; 2022א, מועד 75, שאלה 10; 2021ג, מועד 83, שאלות 11–13; 2021א, מועד 89, שאלה 7).

7.2 מעקב פגיעות והחטאות (G2)

ניסוח	כתיב
פגיעה והחטאה – פגיעה: השורה תקפה (valid = 1) והתגית שווה. החטאה כשהשורה ריקה (״miss valid״) או כשיש בה בלוק אחר (״miss tag״ – התנגשות). בהחטאה הבלוק כולו נטען. (עמ׳ 172–174)	$hit \iff valid \wedge (tag_{line} = tag_{addr})$
החלפה – במיפוי ישיר אין בחירה. ב-K-way: בלוק חדש נכנס לשורה ריקה בקבוצה, ואם אין – **LRU** מוציא את זה שהשתמשו בו לפני הכי הרבה זמן (גם פגיעה מעדכנת את הסדר); FIFO – את זה שנכנס ראשון. (עמ׳ 182–184)	LRU: האחרון **בשימוש** FIFO: האחרון **בכניסה**

תבנית – "כמה פגיעות / מה במטמון בסוף / LRU מול מיפוי ישיר"

1. **לכל כתובת:** בלוק = הכתובת חלקי גודל הבלוק (בבתים), קבוצה = בלוק mod מספר הקבוצות. כתובות של מילים – חלקי מילים בבלוק.
2. **טבלה:** כתובת | בלוק | קבוצה | H/M | עמודה לכל שורה (K עמודות לכל קבוצה) עם התוכן **אחרי** הגישה | מי הוצא.
3. **בכל החטאה בקבוצה מלאה** – מסמנים מי ה-LRU לפי הגישות הקודמות, לא לפי העמודה השמאלית.
4. **ספירה:** פגיעות, החטאות, שיעור החטאה = החטאות חלקי גישות.

דוגמה – 8 בלוקים של מילה, 2-way LRU (4 קבוצות), הכתובות 0, 32, 0, 24, 32, 64, 0 (cache.py, זהה ל-Dinero IV)

address	block	set	H/M	set 0	set 0	set 2	set 2	replaced
0	0	0	M	0				
32	8	0	M	0	8			
0	0	0	**H**	0	8			
24	6	2	M	0	8	6		
32	8	0	**H**	0	8	6		
64	16	0	M	16	8	6		0
0	0	0	M	16	0	6		8

2 פגיעות מתוך 7. במיפוי ישיר על אותו רצף – 0 פגיעות (הבלוקים 0, 8, 16 נלחמים על שורה 0).

שימו לב:

- **פחות בלוקים = יותר "miss tag"** (התנגשויות), לא "miss valid" (2022א, מועד 82, שאלה 11 ; 2022ג, מועד 92, שאלה 14).
- **בדיקת התגית** משווה כל סיבית ומחברת ב-OR; AND היה נותן פגיעות שגויות (2023א, מועד 89, שאלה 12).

באילו מבחנים: (2026ג, מועד 84, שאלה 11 ; 2026א, מועד 96, שאלה 12 ; 2025א, מועד 81, שאלה 14 ; 2022ג, מועד 65, שאלה 14).

7.3 גודל המטמון בסיביות (G3)

ניסוח	כתיב
"גודל המטמון" הוא הנתונים בלבד. **הגודל הכולל** מוסיף לכל שורה תגית וסיבית valid (ו-dirty ב-write-back). (עמ' 176, 179)	$2^n \cdot K \cdot (2^m \cdot 32 + tag + 1\ [+1])$

תבנית – "כמה סיביות בסך הכול / כמה תקורה"

1. **השדות** (7.1): n, m, tag.
2. **שורה אחת:** valid + dirty (אם write-back) + tag + $2^m \cdot 32$.
3. **כפול מספר השורות** $= \#sets \cdot K$.
4. **תקורה:** סך הכול חלקי סיביות הנתונים.

דוגמה – המטמון של 7.1, write-back (cache.py)

valid	dirty	tag	data	per line	lines	total
1	1	20	128	150	256	**38,400**

ב-4-way התגית 22, שורה 152 סיביות, וסך הכול $38912 = 152 \cdot 4 \cdot 64$. במדריך ($n = 12$, $m = 2$, בלי dirty):
$2^{12} \cdot (128 + 16 + 1) = 593920$.

שימו לב:

- **שורה עם תגית ו-64" valid:** שורות של 64 בתים, 531 סיביות לשורה" – גם הנתונים וגם התקורה התקבלו (2025א, מועד 81, שאלה 14).
- **גודל הזיכרון** נקבע מרוחב הכתובת: 2^{28} בתים = $2^6 \cdot 2^7 \cdot 16$ = 128KB; 256MB נתונים (2025א, מועד 87, שאלה 14).

באילו מבחנים: (2026א, מועד 82, שאלה 11 ; 2025ג, מועד 91, שאלה 13 ; 2025א, מועד 94, שאלה 14 ; 2025א, מועד 87, שאלה 14 ; 2025א, מועד 81, שאלה 14 ; 2023א, מועד 89, שאלה 14 ; 2022א, מועד 75, שאלה 9 ; 2021א, מועד 83, שאלה 6).

7.4 ביצועים: AMAT ו-CPI עם החטאות (G4)

ניסוח	כתיב
מחזורי השהיה – גישות לתוכנית כפול שיעור החטאה כפול קנס ההחטאה, **במחזורי שעון**. גישות לפקודה: 1 להבאת הפקודה, ועוד החלק של sw/lw לנתונים. (עמ' 180)	$\text{stall cycles} = \frac{accesses}{program} \cdot miss\ rate \cdot miss\ penalty$
CPI עם החטאות ו-AMAT – CPI הבסיס ועוד ההשהיות לפקודה מכל מקור. זמן גישה ממוצע: זמן פגיעה ועוד שיעור החטאה כפול הקנס. (עמ' 180–181; הספר עמ' 418–420)	$CPI = CPI_{base} + \sum stalls$ $AMAT = hit\ time + miss\ rate \cdot penalty$
רמה שנייה (L2) – החטאה ב-L1 עולה את זמן L2, ורק ההחטאות הגלובליות עולות את זמן הזיכרון. קנס בננו-שניות → מחזורים: כפול התדר. (עמ' 185)	$CPI = CPI_{base} + mr_{L1} \cdot T_{L2} + mr_{global} \cdot T_{mem}$

תבנית – "מה ה-CPI / פי כמה מהיר יותר עם מטמון מושלם / עם L2"

1. **רשימת נתונים** עם יחידות; קנס שנתון בזמן – למחזורים ($ns \cdot GHz$).
2. **טבלה:** מקור | גישות לפקודה | שיעור החטאה | קנס | השהיות לפקודה.
3. **CPI** = בסיס + סכום העמודה האחרונה [cc/ins].
4. **השוואה:** מטמון מושלם – היחס CPI/CPI_{base} (אותו IC ואותו שעון); זמן – $IC \cdot CPI \cdot CCT$.

דוגמה – CPI בסיס 1, קנס 100 מחזורים, החטאת פקודות 2%, נתונים 4%, sw/lw 36% (cache.py)

מקור	גישות לפקודה	שיעור החטאה	קנס	השהיות לפקודה
פקודות	1	0.02	100	2
נתונים	0.36	0.04	100	1.44

$CPI = 1 + 2 + 1.44 = 4.44$ [cc/ins] – פי 4.44 איטי ממטמון מושלם.

- AMAT: פגיעה במחזור אחד, 5% החטאה, קנס 20 – $1 + 0.05 \cdot 20 = 2$ מחזורים.
- L2 (המדריך): 4GHz, זיכרון 400 = 100ns מחזורים, החטאה 2% – $CPI = 9$; עם L2 של 5ns (20 מחזורים) והחטאה גלובלית 0.5% – $1 + 0.02 \cdot 20 + 0.005 \cdot 400 = 3.4$, פי $3.4/2.6 \approx 1.9$.

שימו לב:

- **שיעור החטאת הנתונים** מוכפל רק בחלק של sw/lw (כאן 30%): $1 \cdot 0.02 \cdot 30 + 0.3 \cdot 0.03 \cdot 40 = 0.6 + 0.36 = 0.96$; את ה-CPI הבסיסי לא צריך לדעת (2022א, מועד 75, שאלה 11).
- **שיעור פגיעה ויחידות:** פגיעה 77% היא החטאה 0.23: $AMAT = 2 + 0.23 \cdot 100 = 25$ מחזורים, ובשעון של 1.5ns – 37.5ns (2021א, מועד 83, שאלה 7).

באילו מבחנים: (2026ג, מועד 84, שאלה 10; 2026א, מועד 96, שאלה 10; 2026א, מועד 91, שאלה 12; 2023א, מועד 89, שאלה 13; 2022ג, מועד 65, שאלה 6; 2022א, מועד 75, שאלה 11; 2021א, מועד 89, שאלה 6; 2021א, מועד 83, שאלה 7).

7.5 מדיניות כתיבה (G5)

ניסוח	כתיב
write-through – כל כתיבה הולכת גם לזיכרון הראשי; חוצץ כתיבה (write buffer) חוסך את ההמתנה עד שהוא מתמלא. **write-back** – כותבים רק למטמון ומסמנים dirty; הבלוק נכתב לזיכרון רק כשהוא מוצא. **write-allocate** – בהחטאת כתיבה טוענים את הבלוק; בלי – כותבים ישר לזיכרון. (עמ' 179–180)	write-through: כתיבה לזיכרון בכל sw write-back: כתיבה בהוצאת בלוק dirty

תבנית – "כמה כתיבות לזיכרון / מה במטמון"

1. **טבלה:** גישה (R/W, בלוק) | H/M | התוכן אחרי | dirty | מה הלך לזיכרון – עמודה לכל מדיניות.
2. **write-back:** סופרים כתיבות רק בהוצאת בלוק dirty (ומה שנשאר dirty בסוף, אם שואלים).
3. **write-through בלי allocate:** כל W היא כתיבה לזיכרון, והחטאת W לא טוענת בלוק.

דוגמה – 4 שורות של מילה, מיפוי ישיר, W0, R0, W0, R4, W1 (cache.py, זהה ל-Dinero IV)

גישה	write-back + allocate	write-through בלי allocate
W 0	M, נטען, dirty	M, כתיבה לזיכרון
R 0	H	M, נטען
W 0	H, dirty	H, כתיבה לזיכרון
R 4	M, מוציא את 0 – **כתיבה חזרה**	M, מחליף את 0
W 1	M, נטען, dirty	M, כתיבה לזיכרון
סה"כ	2 פגיעות, כתיבה אחת לזיכרון (ועוד בלוק dirty בסוף)	פגיעה אחת, 3 כתיבות לזיכרון

באילו מבחנים: לא נשאלה מאז (2019א, מועד 82, שאלה 13); מופיעה כשאלת הגדרה.