מערכות הפעלה

3

דיסקים

יחידות מידה

Decimal term	Abbreviation	בסיס דצימלי Value	בסיס בינרי Value	% Larger
kilobyte	KB	10^3	2^{10}	2%
megabyte	MB	10^6	2^{20}	5%
gigabyte	GB	10^9	2^{30}	7%
terabyte	TB	10^{12}	2^{40}	10%

חלקי שניה

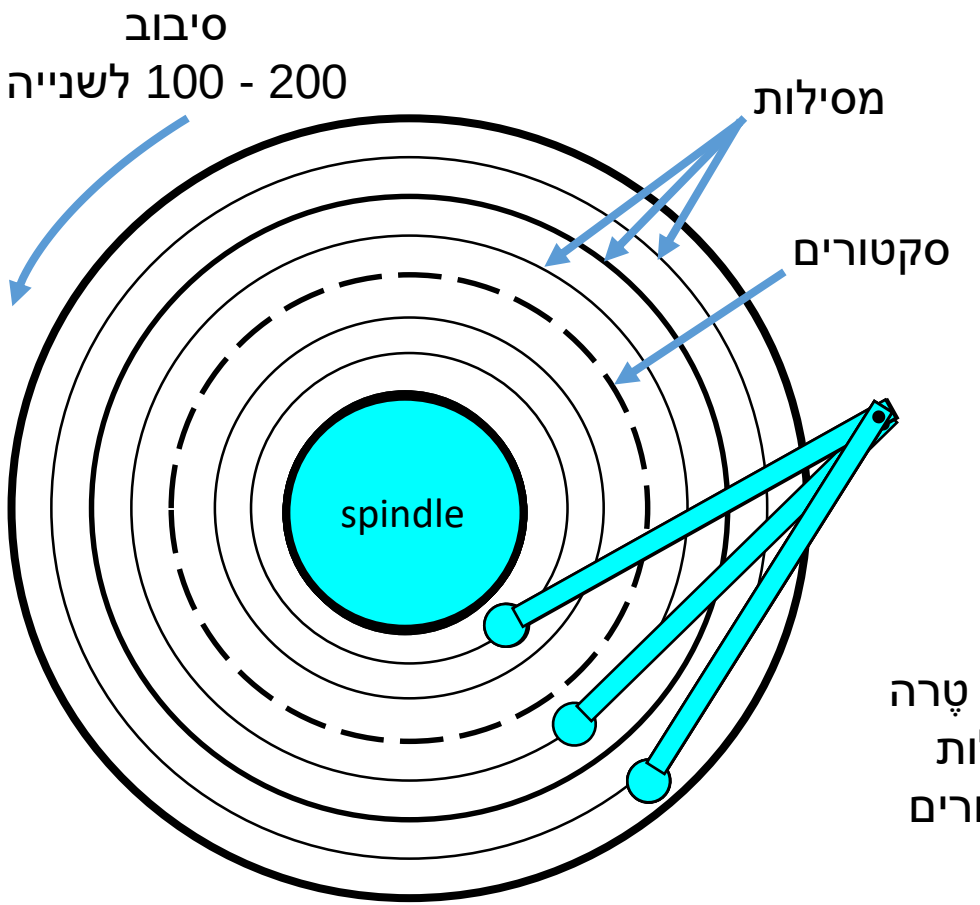
מונח	קיצור	ערך
מילי שנייה	ms	10^{-3} s
מיקרו שניה	µs	10^{-6} s
ננו שניה	ns	10^{-9} s

חזקות של 2

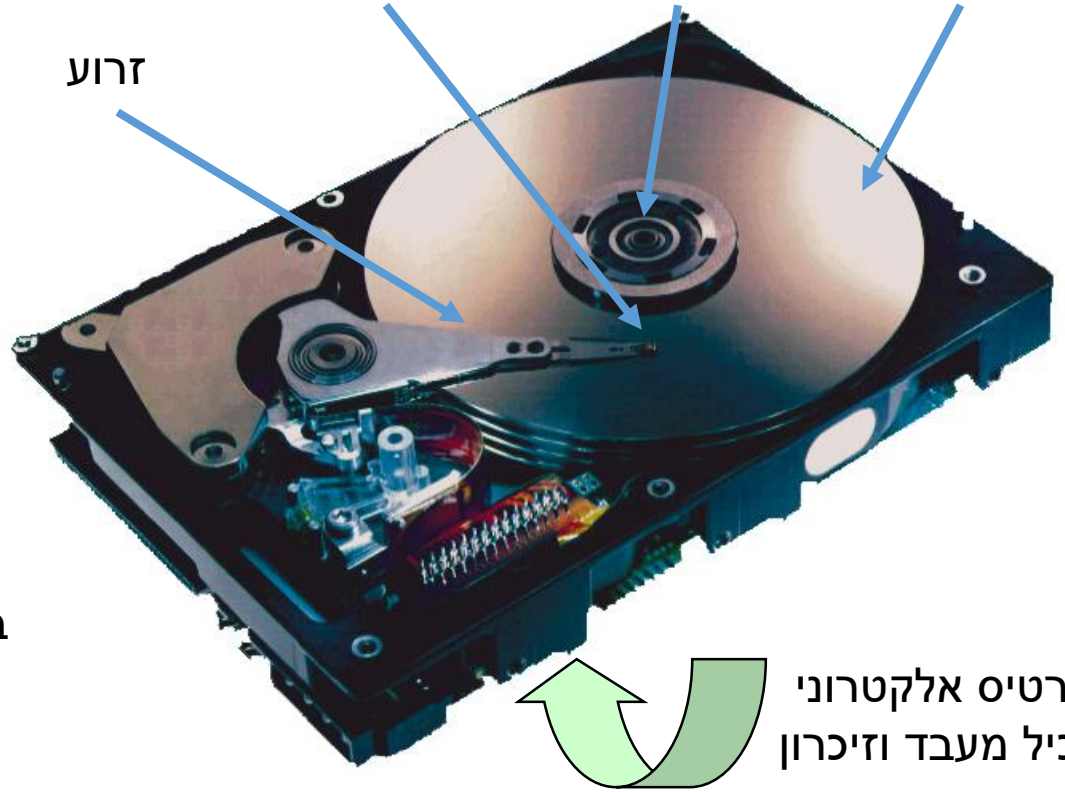
בתים (8 ביטים)	חזקה
512 (0.5K)	2^9
1024 (1K)	2^{10}
4096 (4K)	2^{12}

דיסק מכני

- הדיסק מכיל משטח עגול אחד או יותר מצופה בחומר מגנטי.
- כל משטח מחולק לטבעות הנקראות מסילות, כל מסילה מחולקת לסקטורים, בהם נשמר המידע.
- סיבוב הדיסק ותזוזת הזרוע מאפשרות למקם את ראש הקריאה/כתיבה מעל כל סקטור.
- סקטור הוא יחידת הזיכרון הקטנה ביותר שאפשר לכתוב או לקרוא, גודלו 512 בתים או יותר.



בדיסק בנפח 1 טֶרה
כמיליון מסילות
כאלפיים סקטורים



כרטיס אלקטרוני
מכיל מעבד וזיכרון

חלוקת המסילות לסקטורים

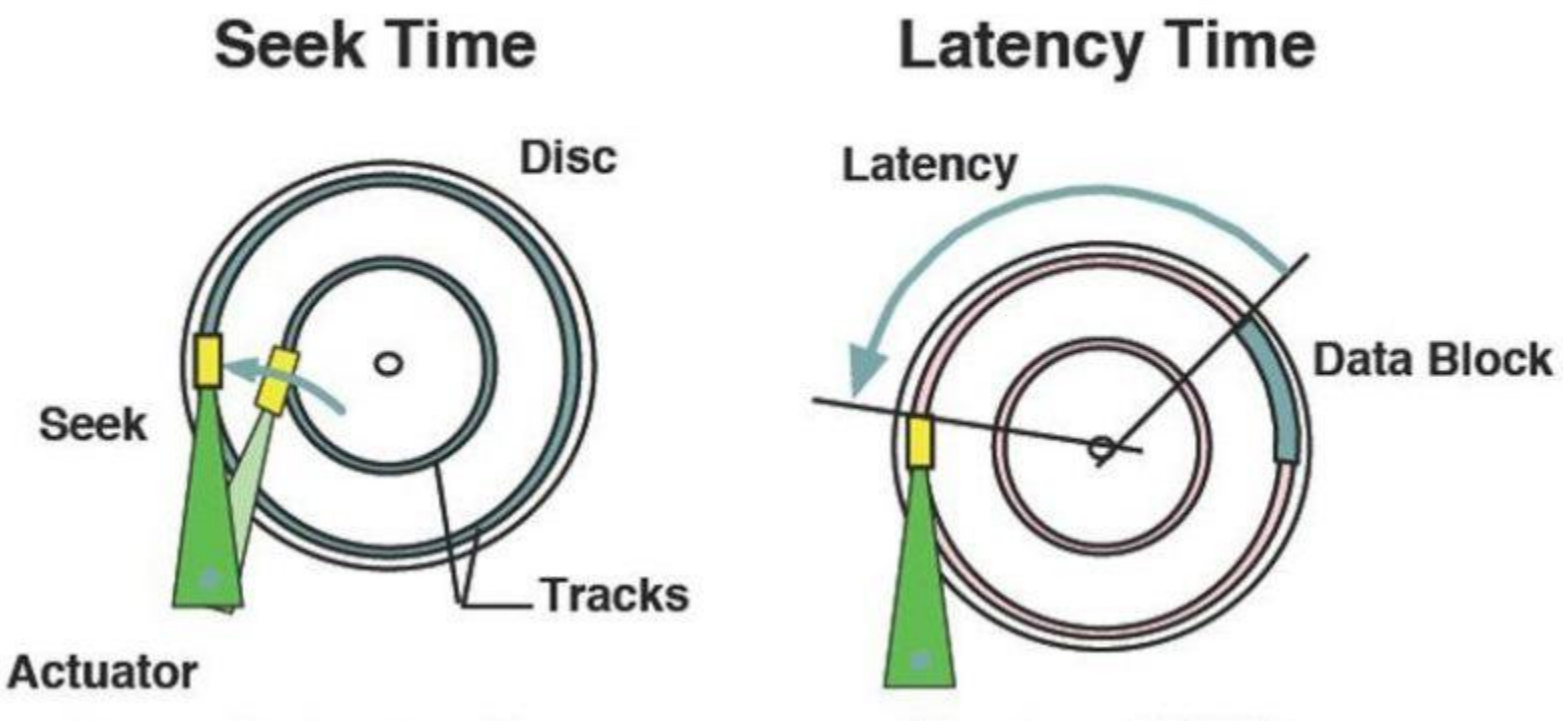
- בתחילת כל סקטור ישנה כותרת שמכילה את המספר של הסקטור.
- בסוף כל סקטור ישנה סיומת שמכילה קוד לבדיקת/תיקון שגיאה.
- ביניהם יש את הנתונים של הסקטור, בדרך כלל בגודל 512 בתים.
- פירמוט הדיסק ברמה נמוכה, מחלק את המסילות לסקטורים ומוסיף את הכותרת והסיומת.
- בדרך כלל דיסק מגיע מהיצרן כשהוא מפורמט.

זמן הגישה לדיסק מכני

- הזמן הממוצע לקריאה או כתיבת סקטור בודד (Access) הוא סכום של שלושה זמנים :

Access = Seek + Latency + Transfer

1. Seek - הזמן שלוקח למקם את הראש מעל המסילה (כמה אלפיות השנייה).
2. Latency - זמן ההמתנה לתחילת הסקטור שיגיע לראש (כמה אלפיות השנייה).
3. Transfer - זמן המעבר של הסקטור מתחת לראש (קטן מאלפית השנייה).



זמן הגישה לדיסק מכני

- זמן הגישה נקבע בעיקר על ידי שני הזמנים הראשונים.
  - אם הבלוקים של המידע מפוזרים על פני הדיסק במסילות שונות, יידרשו שני הזמנים הראשונים לכל בלוק.
  - אם הבלוקים של המידע רציפים בדיסק, יידרשו שני הזמנים הראשונים רק לבלוק הראשון, ליתר הבלוקים יידרש רק הזמן הקצר של מעבר על הסקטורים.
  - מסיבה זו, מערכת ההפעלה קוראת וכותבת לדיסק ביחידות של כמה בלוקים רצופים (4K = 8 סקטורים).
  - בדיסקים מודרניים מערכת ההפעלה קוראת וכותבת למערך של בלוקים לוגיים (LBA) .
  - הדיסק עצמו מחשב את המסילה, המשטח והסקטור של כל בלוק (CHS) .
- זמני גישה לזיכרון ולדיסק:

Memory technology	Typical access time
SRAM semiconductor memory	0.5–2.5 ns
DRAM semiconductor memory	50–70 ns
Flash semiconductor memory	5,000–50,000 ns
Magnetic disk	5,000,000–20,000,000 ns

חישוב זמן הגישה לדיסק

לדיסק יש את הנתונים הבאים:

```
Sector size: 512 bytes              Surfaces: 4
Rotational rate: 10,000 RPM         seek: 5 ms
Number of sectors/track: 1,000
```

כמה זמן ייקח לקרוא קובץ בגודל 1MB (2000*512) מתחילתו ועד סופו, כאשר הבלוקים מסודרים בצורה הגרועה ביותר ובצורה הטובה ביותר.

```
full_rotation = 1/10,000 * 60 * 1000  = 6 ms
latency =  full_rotation * 0.5 = 3 ms
transfer =  full_rotation / 1000 sectors = 0.006 ms
```

הקובץ מכיל 2000 בלוקים, בסידור הגרוע:

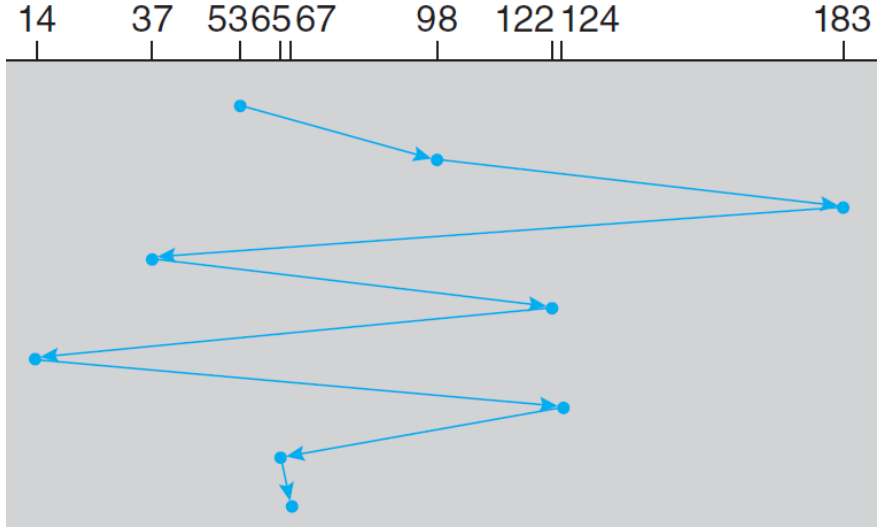
```
(seek + latency + transfer) * 2,000 =
     (5 + 3 + 0.006) * 2,000 = 16,012 ms
```

בסידור הטוב:

```
seek + latency + 2 * full_rotation = 5+3+12 = 20 ms
```

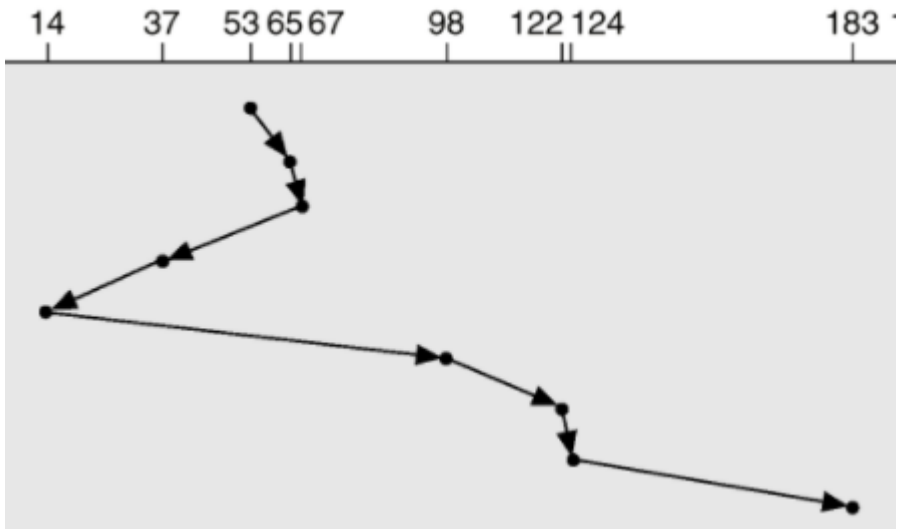
תזמון הדיסק

- בזמן שהדיסק מבצע פעולת קריאה או כתיבה יכולות להגיע למערכת ההפעלה בקשות נוספות לקריאה וכתיבה, הבקשות נכנסות לתור המתנה לדיסק.
  - באיזה סדר תעביר מערכת ההפעלה לדיסק את הבקשות?
- First Come First Served :FCFS algorithm (or FIFO)
  - הראשון שביקש הוא הראשון לקבל.
  - אלגוריתם הוגן אך לא יעיל.
  - נניח שתור הבקשות הוא בלוקים שנמצאים במסילות: 98, 183, 37, 122, 14, 124, 65, 67
  - ראש הקריאה/כתיבה יצטרך לנוע הלוך ושוב על פני הדיסק (הראש נמצא ב- 53):



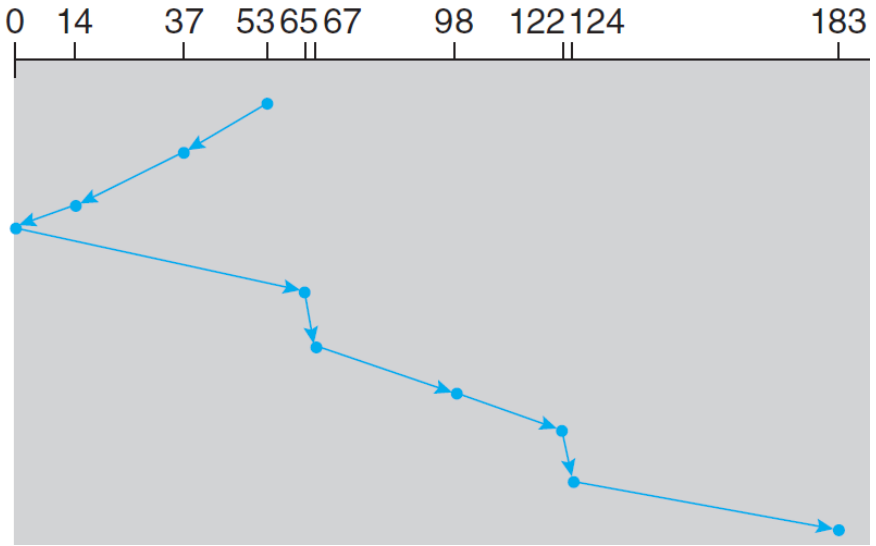
תזמון הדיסק

- Shortest Seek Time First :SSTF algorithm
  - הבקשה הבאה שתועבר לדיסק תהיה זו שנמצאת במרחק החיפוש הקצר ביותר מהבקשה הקודמת.
  - בשיטה זו, בתור הבקשות של הדוגמה, מספר המסילות שהראש ינוע על פניהם הוא שליש ממספר המסילות של השיטה הקודמת.
  - אבל שיטה זו יכולה לגרום **להרעבה** של בקשות שרחוקות מהבקשה הנוכחית, אם יגיע זרם של בקשות קרובות.



תזמון הדיסק

- elevator algorithm: אלגוריתם המעלית
  - הראש מתחיל בקצה אחד ונע לכיוון הקצה השני ובדרך מטפל בבקשות.
  - בקשה שהגיע למסילה שהראש עבר אותה תחכה למעבר הבא.
  - כשהראש מגיע לקצה השני הוא הופך כוון ובדרך מטפל בבקשות.
    - המסילות האמצעיות מקבלות טיפול יותר מהיר.
  - אפשרות אחרת היא חזרה מהירה לקצה הראשון ללא טיפול בבקשות.
    - הוגן לכל המסילות.

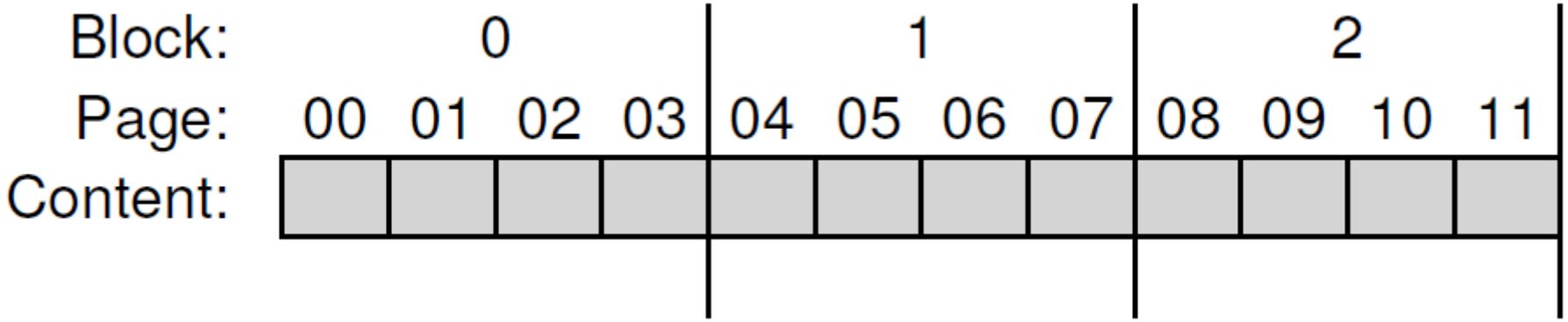


דיסק SSD (solid-state disk)

- לדיסק SSD יש זיכרון מסוג פלאש.
- SSD יותר אמין מדיסק מכני (HDD) מאחר שאין בו חלקים נעים.
- SSD יותר מהיר כי אין זמן חיפוש (seek) או המתנה (latency).
- SSD יותר יקר מדיסק מכני.

חלוקת דיסק SSD

- דיסק SSD מחולק לבלוקים והבלוקים מחולקים לדפים.
- הבלוקים הם בגודל 128 KB או 256 KB, והדפים בגודל 4 KB.



- כדי לכתוב לדף שבתוך הבלוק, צריך למחוק את כל הבלוק.
- אם מוחקים בלוק הרבה פעמים (10,000 – 100,000) הבלוק נשחק ולא שומר מידע.

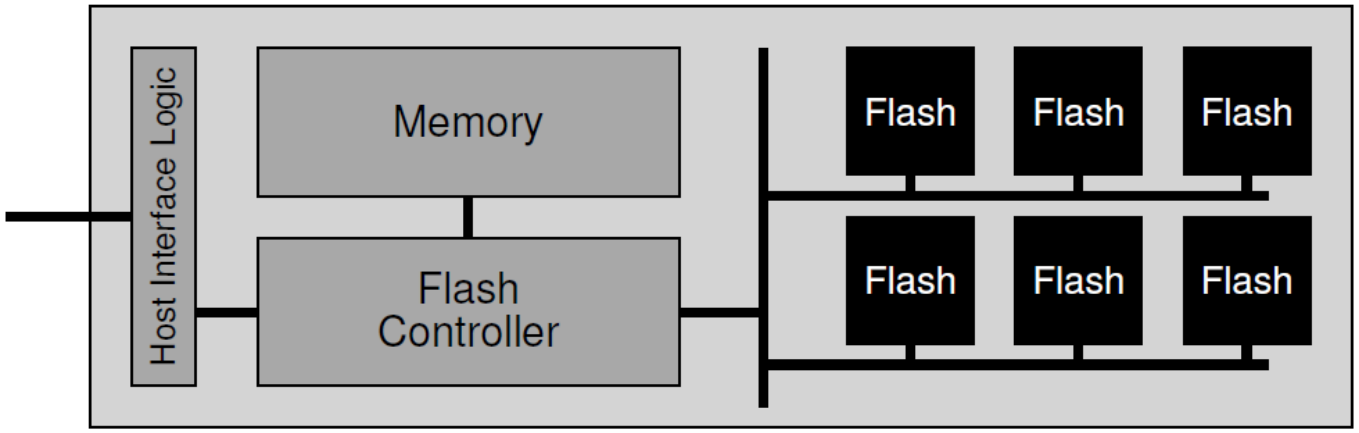
פעולות בדיסק SSD

- **קריאת דף:** ניתן לקרוא כל דף בשלמותו, אם מציינים את מספר הדף.
  - פעולת קריאה היא מהירה - עשרות מיקרו-שניות (מיליוניות השנייה).
- **מחיקת בלוק:** לפני כתיבה לדף צריך למחוק את כל הבלוק.
  - יש צורך להעתיק דפים אחרים בבלוק שיש בהם תוכן למקום אחר.
  - פעולות מחיקה היא איטית – כמה מילישניות (אלפיות השנייה).
- **כתיבת דף:** לאחר מחיקת הבלוק אפשר לכתוב בו דפים.
  - פעולת כתיבה יותר איטית מקריאה - מאות מיקרו-שניות (מיליוניות השנייה).
- מאפייני שלושה דיסקים:

Device	Read (μs)	Program (μs)	Erase (μs)
SLC	25	200-300	1500-2000
MLC	50	600-900	~3000
TLC	~75	~900-1350	~4500

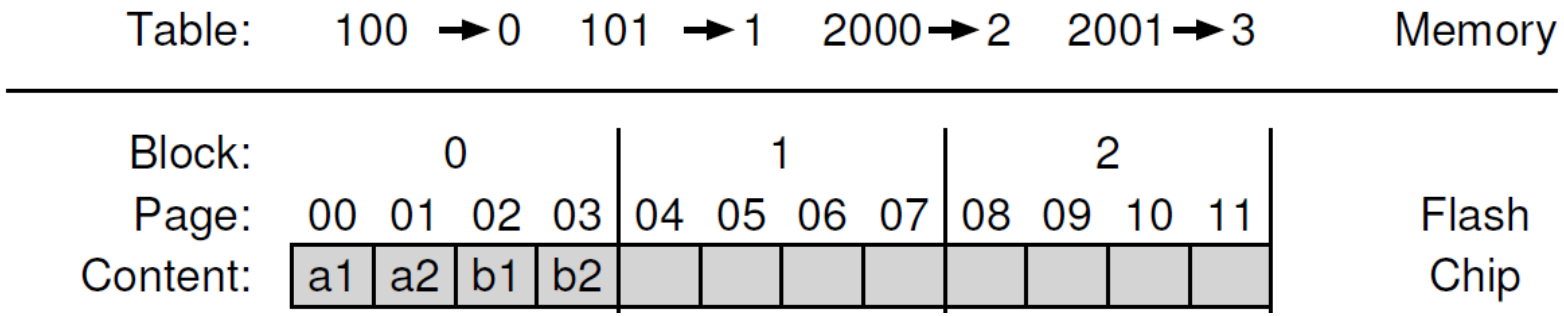
מיפוי דפים לוגיים לפיסיים - FTL (flash translation layer)

- בנוסף לרכיבי זיכרון מסוג flash, דיסק SSD מכיל זיכרון RAM ובקר (Controller) שמבצע תוכנה שנקראת FTL (flash translation layer)
- FTL מקבל פקודות ממערכת ההפעלה לקריאה וכתיבה של דפים לוגיים ומתרגם את הפקודות לקריאה, מחיקה וכתיבה לדיסק של דפים פיסיים.
- מיפוי ישיר של דפים לוגיים לפיסיים אינו רצוי:
  - עבור כל כתיבה לדף יש צורך לקרוא ולשמור את כל הדפים שבאותו בלוק, למחוק את הבלוק, לכתוב את הדף המעודכן ולהחזיר את יתר הדפים.
  - זמן הביצוע עשוי להיות יותר איטי מדיסק.
  - אם ישנו דף שמעדכנים אותו הרבה פעמים, יהיו הרבה מחיקות ושחיקה של הבלוק.



מיפוי בשיטת כתיבה ללוג

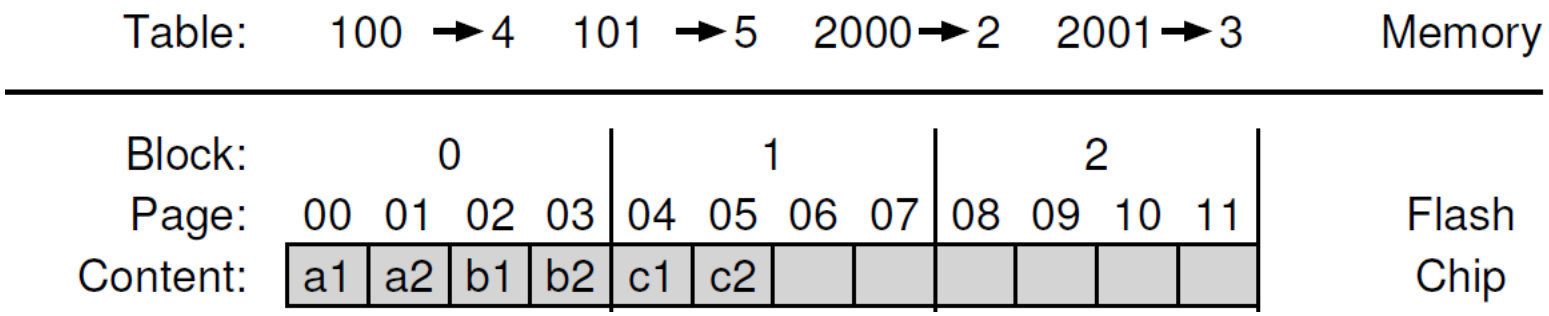
- בשיטת כתיבה ללוג, כתיבה של דף לא מחליפה את התוכן של הדף הקודם, במקום זה כותבים לדף אחר מתוך הדפים הפנויים.
- כדי שיהיה אפשר למצוא אחרי כן את הדף שנכתב, יש צורך במיפוי הדפים הלוגיים לדפים פיסיים.
  - דוגמה, נניח דיסק חדש ומערכת ההפעלה כתבה לדפים 100, 101, 2000, 2001, הם ימופו לדפים 0, 1, 2, 3 בבלוק 0. המיפוי נשמר בזיכרון ה- RAM (וגם בדפים):



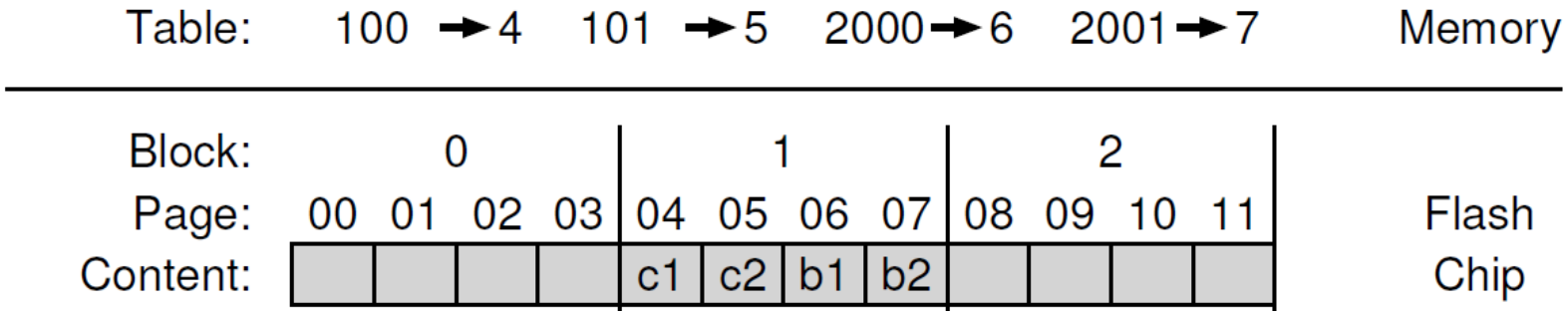
- אם עדכנו את הדפים הלוגיים 100, 101 אזי הם ייכתבו לדפים הפיסיים 4, 5 (בבלוק 1 שנמחק ומוכן לכתיבה).
- אם כן, אין צורך למחוק את כל הבלוק עבור כתיבה של דף ולכן הכתיבה תהיה מהירה.
- וכן המחיקות יהיו מפוזרות על פני הדיסק ותהיה פחות שחיקה.

Garbage Collection

- בשיטת כתיבה ללוג, הדפים הקודמים לא מכילים תוכן מעודכן והם מצורפים לרשימת הדפים הפנויים.
- בהמשך לדוגמה הקודמת, נניח שעדכנו את הדפים הלוגיים 100, 101 אזי הם יכתבו לדפים הפיסיים 4, 5 והמיפוי יעודכן:

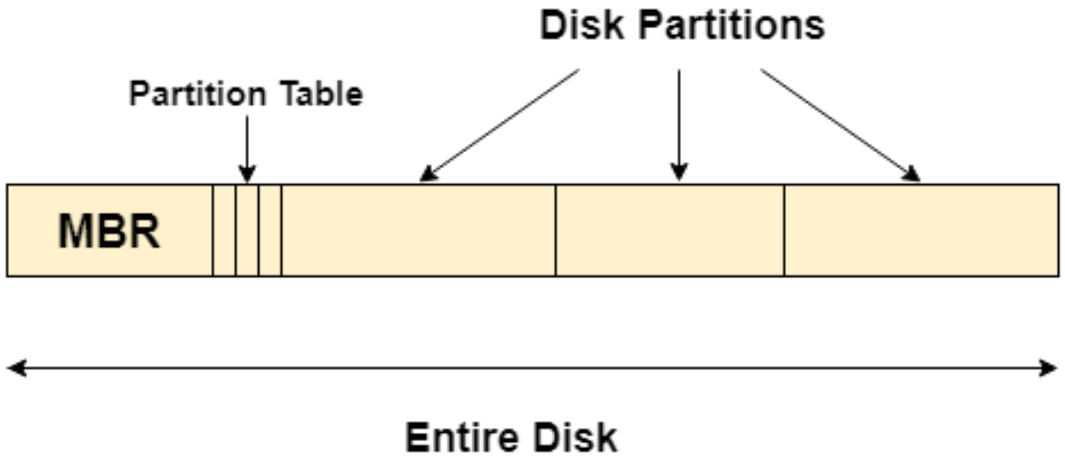


- כדי להשתמש בדפים הקודמים יש צורך למחוק את כל הבלוק של אותם דפים (ולפני כן להעתיק דפים אחרים שהם בשימוש ללוג):



חלוקת דיסק למחיצות - Disk partitions

- אפשר לחלק את דיסק למחיצות.
  - כל מחיצה היא דיסק לוגי (נראה למערכת ההפעלה כדיסק רגיל)
  - בכל מחיצה אפשר ליצור מערכת קבצים נפרדת.
- מחיצה יכולה להכיל:
  - file system - מערכת קבצים.
  - swap – אזור בדיסק שמערכת ההפעלה שומרת בו קטעי זיכרון כאשר הזיכרון מלא.
  - raw device – אזור בדיסק ללא מערכת קבצים, האפליקציה נגשת ישירות לבלוקים, יכול לשמש במערכות בסיסי נתונים.

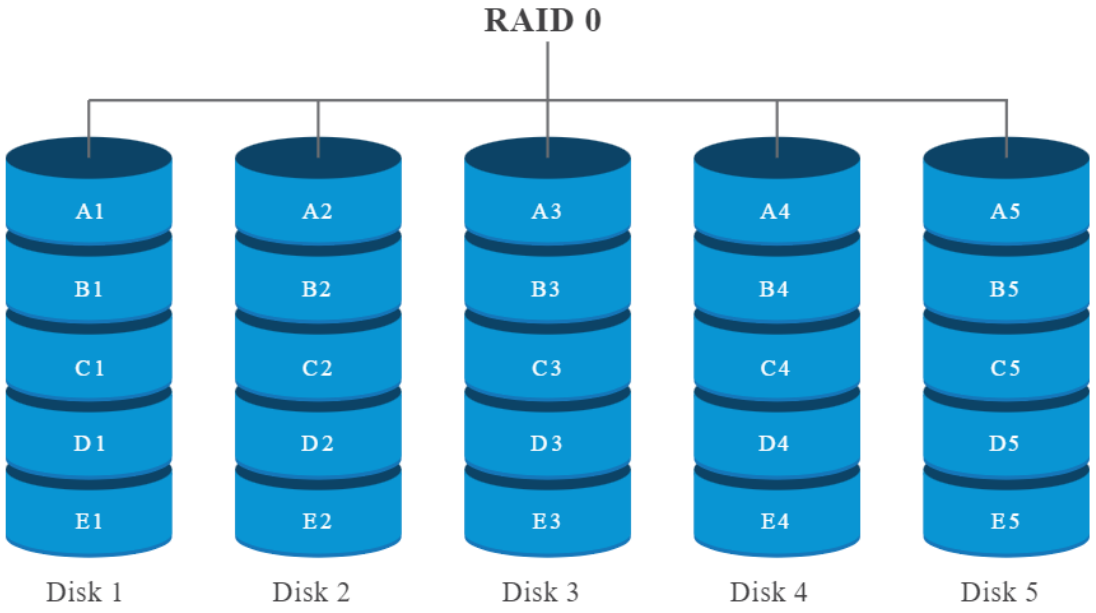


RAID - Redundant Arrays of Inexpensive Disks

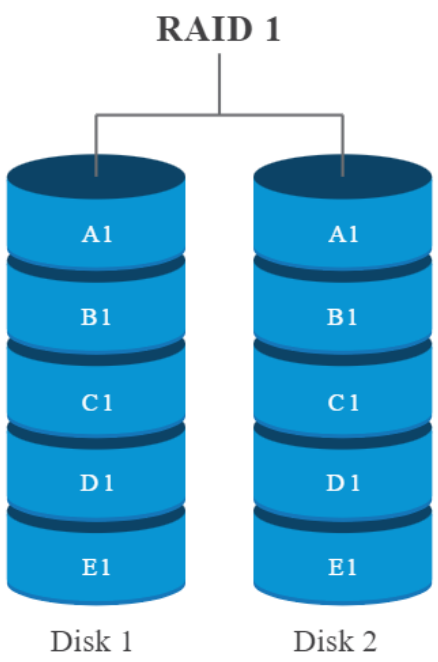
- ראינו שאפשר לחלק דיסק פיזי לדיסקים לוגיים, כעת נראה איך אפשר לחבר כמה דיסקים פיזיים לדיסק לוגי אחד (volume).
- היתרונות בחיבור דיסקים:
  - שיפור ביצועים - אפשר לגשת לדיסקים במקביל.
  - אמינות – אם יש תקלה בדיסק, אפשר לשחזר את הנתונים באמצעות מידע יתיר (redundant) שנשמר ביתר הדיסקים.
- RAID היא שיטה לחיבור כמה דיסקים לדיסק אחד.
- שיטת RAID כוללת כמה אפשרויות לחיבור דיסקים:
  - RAID 0, … , RAID 6
  - להלן נראה את האפשרויות הנפוצות.

RAID 0 - Striping (פסים)

- ראינו שמערכת ההפעלה רואה את הדיסק כמערך של בלוקים.
- בשיטת RAID 0 מחלקים את הדיסקים לפסים, בלוק 0 הוא בדיסק הראשון, בלוק 1 בדיסק הבא וכן הלאה.
  - ביצועים: אפשר לגשת לדיסקים במקביל, נותן את הביצועים הטובים ביותר של RAID. לדוגמה, אפשר לקרוא או לכתוב קובץ בגודל שני בלוקים בשני דיסקים בו זמנית.
  - נפח: צרוף הדיסקים לא מפחית מהנפח הכולל של הדיסקים.
  - אמינות: אין יתירות, אם יש תקלה בדיסק אחד, המידע שעל כל הדיסקים משתבש, יותר גרוע מדיסקים ללא RAID.



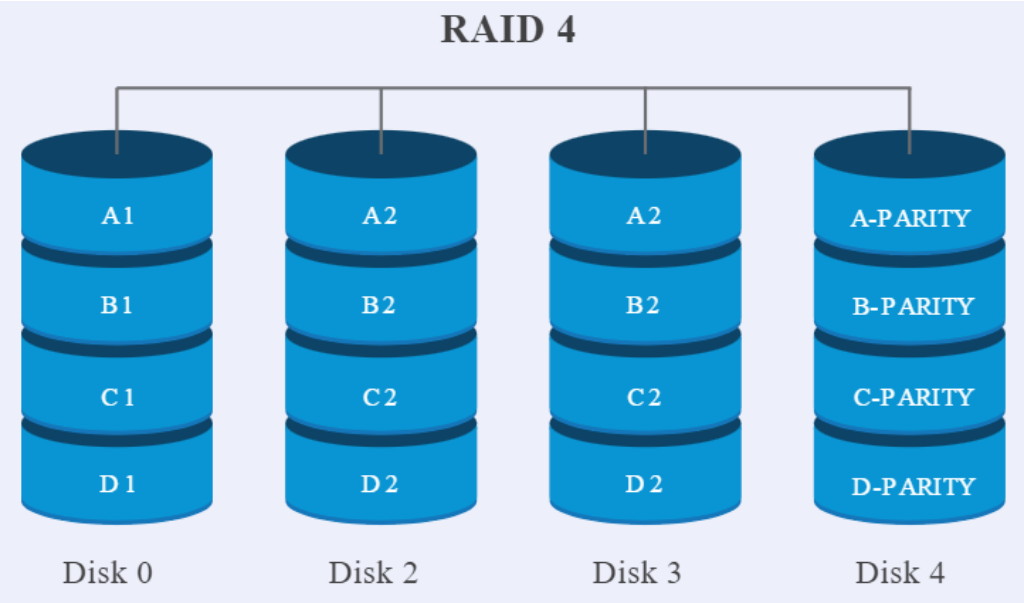
RAID 1 – Mirroring (שכפול)



- בשיטת RAID 1 כל הנתונים נכתבים לשני דיסקים (ראי).
  - ביצועים: קריאה היא יותר מהירה כי אפשר לקרוא משני הדיסקים במקביל.
  - נפח: בשיטה זו אפשר לנצל רק חצי מנפח הדיסקים.
  - אמינות: אם יש תקלה בדיסק, אפשר לקרוא מהדיסק השני.

RAID 4 – Parity (זוגיות)

- בשיטת RAID 4 מחלקים את הדיסקים לפסים.
- בדיסק האחרון נשמר מידע (זוגיות) שמאפשר שחזור הנתונים במקרה של תקלה באחד מהדיסקים האחרים.
  - ביצועים: משפר את הביצועים, אפשר לגשת לדיסקים במקביל.
  - נפח: דיסק אחד משמש לזוגיות ולכן הנפח לשמירת נתונים הוא כמספר הדיסקים פחות אחד.
  - אמינות: אם יש תקלה בדיסק **אחד**, אפשר לשחזר את הנתונים מיתר הדיסקים.



זוגיות (Parity)

- זוגיות היא פונקציה שמחשבים על הבלוקים של הנתונים של כל פס ושומרים את התוצאה בבלוק שבדיסק האחרון.
  - הפונקציה שמחשבים היא XOR על הביט הראשון של כל הבלוקים של הפס ושמירת התוצאה בביט הראשון של בלוק הזוגיות, XOR על הביט השני ושמירת התוצאה בביט השני של בלוק הזוגיות, וכן הלאה עבור כל ביט בבלוק.
  - התוצאה של XOR היא 0 אם מספר הביטים שהם 1 הוא זוגי, התוצאה היא 1 אם מספר הביטים שהם 1 הוא אי-זוגי.
  - יוצא שמספר הביטים שהם 1 בכל הבלוקים כולל בבלוק הזוגיות הוא זוגי עבור כל ביט.
- בדוגמה הבאה מספר הדיסקים הוא חמש ובכל בלוק ארבעה ביטים:

Block0	Block1	Block2	Block3	Parity
00	10	11	10	11
10	01	00	01	10

חישוב הזוגיות

- בכל כתיבה של בלוק נתונים יש צורך לעדכן את בלוק הזוגיות.

1. אפשרות אחת היא לקרוא את כל הבלוקים של הפס, לחשב את הזוגיות ולכתוב בבלוק הזוגיות.
2. אפשרות יותר יעילה היא להשוות את הביטים (בהתאמה) של הבלוק שרוצים לכתוב לתוכן הקודם שלו.
   - אם הביטים שווים אזי ביט הנתונים לא השתנה ולכן אין צורך לשנות את ביט הזוגיות.
   - אם הביטים שונים צריך להפוך את ביט הזוגיות.
   - אפשר לבצע את החישוב באופן הבא:

$$P_{new} = (C_{old} \oplus C_{new}) \oplus P_{old}$$

זהויות עבור XOR:

$x \oplus x = 0$

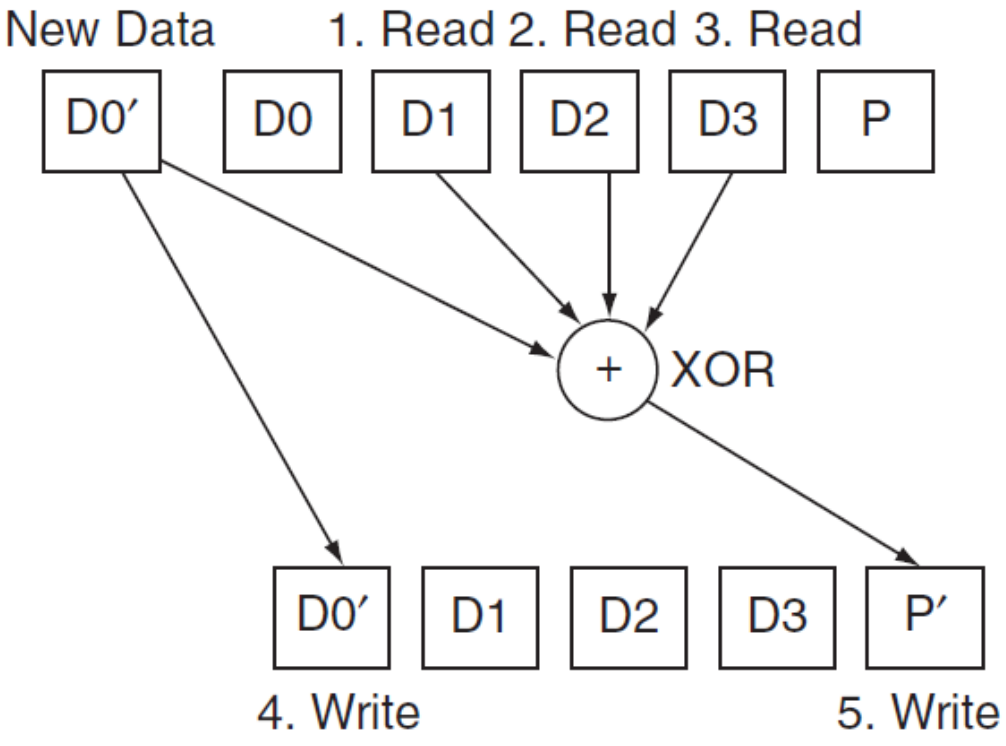
$x \oplus x' = 1$

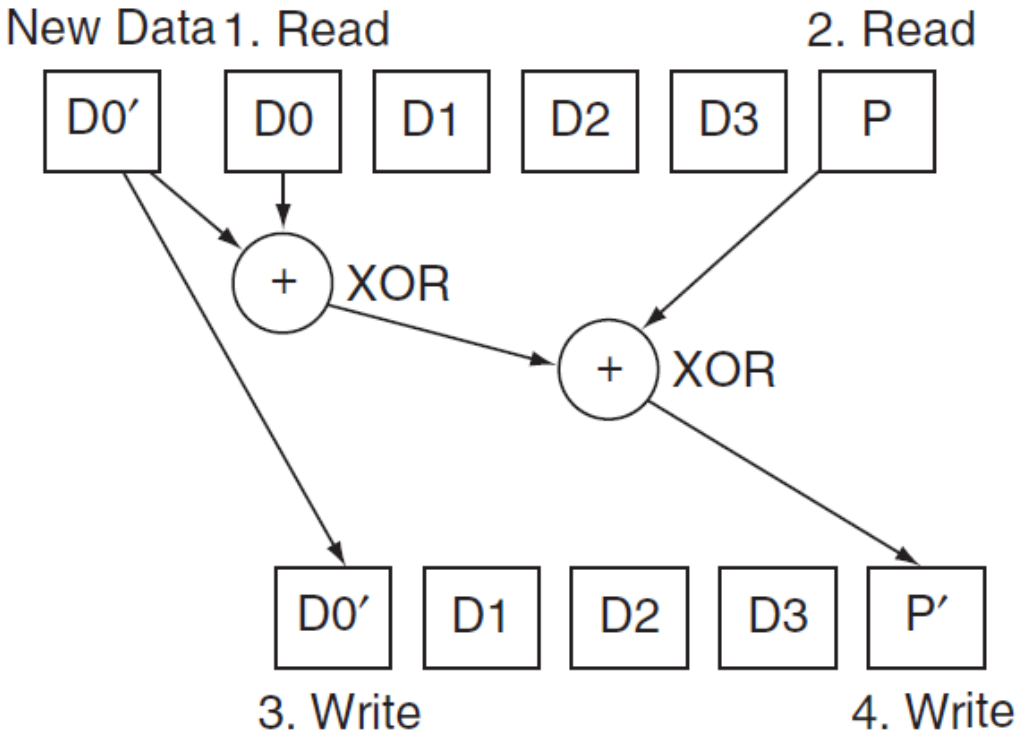
$x \oplus 0 = x$

$x \oplus 1 = x'$

חישוב הזוגיות

- הציור הבא מדגים את שתי האפשרויות:





שחזור נתונים

- אם ארעה תקלה בדיסק, כדי לשחזר את הנתונים בדיסק חדש, נבצע עבור כל בלוק, XOR של כל הדיסקים הנותרים כולל דיסק הזוגיות.
- הסבר:
  - ראינו שבכל הדיסקים כולל דיסק הזוגיות מספר הביטים שהם 1 הוא זוגי.
  - פעולת XOR על הדיסקים הנותרים תיתן 1 אם מספר הביטים הוא אי זוגי, ואז נכתוב בדיסק החדש 1 ונקבל שוב מספר זוגי.
  - פעולת XOR על הדיסקים הנותרים תיתן 0 אם מספר הביטים הוא זוגי, ואז נכתוב 0 בדיסק החדש כי מספר הביטים הוא כבר זוגי.

RAID 5 – Rotating Parity - זוגיות בסבב על כל הדיסקים

- בשיטת RAID 4, בכל כתיבה של בלוק לדיסק כלשהו צריך לעדכן את הבלוק שבדיסק הזוגיות.
- זה הופך את דיסק הזוגיות לצוואר בקבוק בכתיבות לדיסק.
- בשיטת RAID 4, כדי לשפר את הביצועים, הבלוקים של הזוגיות מפוזרים בסבב על כל הדיסקים.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
*4	5	6	7	+P1
8	9	10	11	P2
12	*13	14	15	+P3