ארגון המחשב ומערכות הפעלה

אביב תשפ"ד

תרגול 2 – בתים וסיביות

תרגול קודם

לוגיקה פסוקית

- מה זה פסוק?
- קשרים לוגיים
- התמרות בין אופרטורים
  - חוקי דה-מורגן
  - XOR באמצעות NOT, OR, AND

מעברי בסיסים

- מעבר מבסיסים שונים לבסיס 10
- מעבר מבסיס 10 לבסיסים שונים- אלגוריתם
- בסיס הקסדצימלי (בסיס 16)

והיום... סיביות ובתים (bits and bytes)

- אפשר לבנות מחשב באיזה בסיס שנרצה, הבעיה שבייצוג שהוא לא בינארי הרבה יותר קשה לחשב, לאגור מידע ולהעביר מידע
- כל מידע שהמחשב מעבד- ניתן לייצג באמצעות 0 ו-1 (בבסיס 2)
- נזכור- 1 בית (byte) = 8 סיביות (bits)
- בהרצאה ראינו שלכל מחשב יש רוחב מילה - הגודל המוקצה למידע שמיוצג ע"י integer (כולל הכתובת).
- פעם מחשבים היו ברוחב 4 בתים (32 סיביות), ובאשר נהפך לצפוף - כיום עובדים עם 8 בתים (64 סיביות)

לכל מחשב יש סדר שונה לארגון הזיכרון

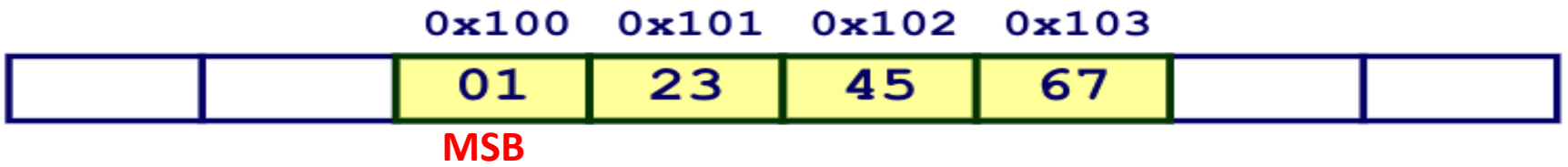
הזיכרון מסודר לפי בתים- לכל בית יש כתובת משלו (מעין מערך ענקי)

נסתכל על המספר הבא: 0x01234567

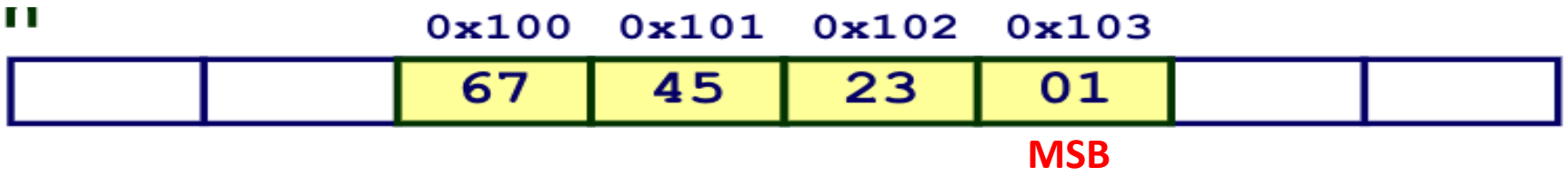
כל ספרה תופסת 4 ביטים ועל כן 2 ספרות מהוות בית אחד (8 ביטים).

אפשר לסדר את הבתים בשני אופנים:

1. ה MSB בכתובת הראשונה - **Big Endian**



2. ה MSB בכתובת האחרונה - **Little Endian**



לא ניתן להסתמך על ארגון הזיכרון אם מייבאים קוד ממעבדים שונים!

אופרטורים

- בשפת C- 0 הוא false ובל מספר אחר הוא true
- **אופרטורים לוגיים** על true או false -
  - ! הוא not לוגי
  - && הוא and לוגי
  - || הוא or לוגי
- **אופרטורים על ביטים (bitwise operators):**

and **&**	or **|**	xor **^**	not **~**
זיזות **<< \ >>** (בהמשך התרגול)	חיבור \ חיסור (בתרגול הבא)		

אופרטורים

נבחן את ההבדל בין שני הסוגים (לוגי מול bitwise).

נסתכל על Not:

$$!13 = !true = false \quad \text{(לוגי)}$$

$$\sim 13 = \sim(1101) = 0010 = 2 \quad \text{(bitwise)}$$

התוצר הוא שונה (בוליאני או מספרי)!

אופרטורים

נסתכל על And:

(לוגי) $(13 \;\&\&\; 2) = (true \;\&\&\; true) = true$

(bitwise) $(13 \;\&\; 2) = (1101 \;\&\; 0010) = 0000 = 0$

פעולות זיזה (shifting operations)

- ישנם שני סוגי זיזות:
  - זיזה שמאלה
  - זיזה ימינה
    - לוגית
    - אריתמטית

זיזה שמאלה x<<y

מזיזים את הווקטור x ב-y סיביות שמאלה

← זורקים את y הסיביות משמאל ומוסיפים אפסים מימין

דוגמה: x=0000 1101, y=3

000**01101** << 3 → **01101**000

אם חושבים על זה- הדבר שקול לפעולה $x * 2^y$ (אם אין גלישה ואין שינוי סימן +-)

למה? נבצע x<<3 עבור $x = x_7*2^7+x_6*2^6+...+x_0*2^0$

$x<<3=x_7*2^{10}+x_6*2^9+...+x_0*2^3+0*2^2+0*2^1+0*2^0$

קיבלנו אותם מקדמים כשחזקותיהם גדולות ב-3

זיזה ימינה x>>y

מזיזים את הווקטור x ב-y סיביות ימינה

← זורקים את y הסיביות בימין, משמאל קורים אחד מהמקרים הבאים:

1. **<u>זיזה לוגית</u>**- אם x הוא unsigned (חיובי או 0 בלבד)- ממלאים בשמאל ב-0-ים
2. **<u>זיזה אריתמטית</u>**- אם x הוא signed (חיוביים ושליליים)- ממלאים בשמאל בסיבית האחרונה- יכול להיות 0-ים או 1-ים (למה?)

דוגמה: x=10100010, y=2

<u>זיזה לוגית-</u> 10100010 >> 2 = 00101000

<u>זיזה אריתמטית-</u> 10100010 >> 2 = 11101000

אם חושבים על זה- הדבר שקול ל $\left\lfloor \frac{x}{2^y} \right\rfloor$

למה?

זיזה ימינה x>>y

נבצע זיזה לוגית x>>2 עבור $x=x_7*2^7+...+x_1*2^1+x_0*2^0$

קיבלנו: $x>>2 = x_7*2^5+...+x_2*2^0$

נשווה לחלוקה ב-2^2:

קיבלנו: $x / 2^2 =x_7*2^5+...+x_2*2^0+x_1*2^{-1}+x_0*2^{-2}$

זו אותה התוצאה, אחרי שמעגלים את השבר מטה

בואו נבנה פונקציה

אנחנו רוצים לחשב את הערך של $2^x + 4$ בהנתן x

אפשר לכתוב 2 פונקציות שונות:

<u>עם לולאות</u>

```
int pow2plus4( int x )
{


}
```

בואו נבנה פונקציה

```
int pow2plus4(int x)
{
    Int temp=1;
    for (int i=0; i<x; i++)
    {
        temp=temp*2;
    }
    temp=temp+4;
    return temp;
}
```

מה הסיבוכיות? O(x)

בואו נבנה פונקציה

<u>בלי לולאות</u>

נשים לב לביטוי 2^x שניתן לכתוב אותו $1 * 2^x$

מזכיר לנו משהו? זיזה שמאלה!

```
int pow2plus4( int x )
{
return (1<<x)+4;
}
```

מה הסיבוכיות? O(1)

תזכורת מההרצאה

טווחים של ערכי בתים:

בסיס בינארי- $0000\ 0000_2\ to\ 1111\ 1111_2$

בסיס דצימלי- $0_{10}\ to\ 255_{10}$

בסיס הקסדצימלי- $00_{16}\ to\ FF_{16}$

Decimal (Base 10)	Binary (Base 2)	Hexadecimal (Base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

תרגיל 1

0x503C + 0x8

0x503C – 0x40

0x503C + 64

0x50EA – 0x503C

*אתר שימושי:

https://madformath.com/calculators/digital-systems/digital-systems-calculators

תרגיל 1 - פתרון

0x503C + 0x8 = 0x5044

0x503C – 0x40 = 0x4FFC

0x503C + 64 = 0x507C

0x50EA – 0x503C = 0x00AE

תרגיל 2

X		X<<3		X>>2 Logical		X>>2 Arithmetic	
Hex	Binary	Binary	Hex	Binary	Hex	Binary	Hex
0xC3							
0x75							
0x87							
0x66							

תרגיל 2 - פתרון

X		X<<3		X>>2 Logical		X>>2 Arithmetic	
Hex	Binary	Binary	Hex	Binary	Hex	Binary	Hex
0xC3	11000011	00011000	0x18	00110000	0x30	11110000	0xF0
0x75	01110101	10101000	0xA8	00011101	0x1D	00011101	0x1D
0x87	10000111	00111000	0x38	00100001	0x21	11100001	0xE1
0x66	01100110	00110000	0x30	00011001	0x19	00011001	0x19