נספח עבודה

הקדמה:

כזכור לכל משתנה מטיפוס פרימיטיבי יש הקצאה של n ביטים בזיכרון. עבור boolean, char, int ו – double יש הקצאה של 1, 16, 32 ו – 64 ביטים בזיכרון בהתאמה. נתמקד במשתנה מטיפוס int עם הקצאה של n=32 ביטים. הקצאה זאת מאפשרת לנו לייצג 2^{32} ביטויים שונים הנבדלים על ידי רצף הביטים, לדוגמא ניתן לייצג את הטווח $[0,2^{32}-1]$ המכיל מספרים שלמים אי-שליליים מבסיס 10.

ברירת המחדל של int בשפת JAVA היא שרצפי הביטים השונים מייצגים מספרים שלמים הן שליליים והן אי-שליליים. שיטת ייצוג זו נקראת **שיטת המשלים ל-2** (Two's complement) עליה נפרט בהמשך. בשיטה זו הביט במיקום השמאלי ביותר (most significant bit–MSB) מציין את הסימון, כלומר אם המספר בבסיס 10 הוא אי-שלילי או שלילי. כאשר הביט השמאלי ביותר הוא 1 המספר הוא שלילי, וכאשר הוא 0 המספר הוא מספר אי-שלילי. לפי האיור הבא:

ייצוג בינארי עבור n = 32 (int)				**ייצוג בבסיס 10**
01111111	11111111	11111111	11111111	2147483647
01111111	11111111	11111111	11111110	2147483646
01111111	11111111	11111111	11111101	2147483645
⋮				⋮
00000000	00000000	00000000	00000011	3
00000000	00000000	00000000	00000010	2
00000000	00000000	00000000	00000001	1
00000000	00000000	00000000	00000000	0
11111111	11111111	11111111	11111111	-1
11111111	11111111	11111111	11111110	-2
11111111	11111111	11111111	11111101	-3
11111111	11111111	11111111	11111100	-4
⋮				⋮
10000000	00000000	00000000	00000010	-2147483646
10000000	00000000	00000000	00000001	-2147483647
10000000	00000000	00000000	00000000	-2147483648

אמנם אנו רגילים להשתמש ב- int עם n = 32 עבור ייצוג מספרים שלמים בבסיס 10, אך באופן כללי ניתן לייצג מספרים שלמים בבסיס 10 **בשיטת המשלים ל-2** גם עבור n = 8, n = 3 ואף n=1024.

כאשר אנו נעזרים ב n כלשהו לייצג מספרים בבסיס 10 בשיטת המשלים ל -2 נוכל להגדיר את טווח הייצוגים באופן הבא: $[-2^{n-1}, 2^{n-1} - 1]$.

למעשה עם שימוש ב- n שונים ניתן לייצג את אותו מספר בבסיס 10 רק עם רשימת ביטים באורך שונה, בתנאי שהוא נמצא בטווח של ה – n בו הוא מוגדר. לדוגמא:

$n = 4\,, [-2^3, 2^3 - 1]$

$n = 8\,, [-2^7, 2^7 - 1]$

$n = 32\,, [-2^{31}, 2^{31} - 1]$

<u>*n=32*</u>   <u>*n=8*</u>   <u>*n=4*</u>

$$(5)_{10} = (00000000|00000000|00000000|00000101)_2 = (00000101)_2 = (0101)_2$$

$$(0)_{10} = (00000000|00000000|00000000|00000000)_2 = (00000000)_2 = (0000)_2$$

$$(1000033)_{10} = (00000000|00001111|01000010|01100001)_2 = \cancel{(01100001)_2} = \cancel{(0001)_2}$$

$$(-5)_{10} = (11111111|11111111|11111111|11111011)_2 = (11111011)_2 = (1011)_2$$

$$(-20)_{10} = (11111111|11111111|11111111|11101100)_2 = (11101100)_2 = \cancel{(1100)_2}$$

$$(-1000033)_{10} = (11111111|11110000|10111101|10011111)_2 = \cancel{(10011111)_2} = \cancel{(1100)_2}$$

מכאן ניתן לראות שלכל מספר בבסיס 10 יש **ייצוג בינארי מינימאלי** בביטים. כלומר את 5 ניתן לייצג עם $n \geq 4$ בלבד בשיטת המשלים ל-2. אם היינו מנסים לייצג את 5 עם $n = 3$ היינו מקבלים את הייצוג הבינארי (101) שהייצוג העשרוני שלו הוא -3 בשיטת המשלים ל -2.

<u>ייצוג בינארי מינימאלי:</u>

לאחר שראינו ביטוי של ייצוג בינארי מינימאלי מהדוגמא הקודמת נפנה כעת להגדיר את המושג. באופן פורמאלי, **ייצוג בינארי מינימאלי** הוא הייצוג של מספר בבסיס 10 בייצוג בינארי כך שה – n (מספר הביטים) איתו הוא מיוצג הינו הקטן ביותר.

לדוגמא את הספרה 5 ניתן לייצג על ידי ייצוגים בינארים שונים n=4,8,32, כך ש n=4 הוא הקטן מביניהם :

$$(5)_{10} = (00000000|00000000|00000000|00000101)_2 = (00000101)_2 = (0101)_2$$

בדוגמא הקודמת הזכרנו שלא יתכן n=3. הסיבה לכך היא שב- n=3 אנו נאלצים להשמיט את הביט השמאלי ביותר (MSB), שכזכור מורה לנו על סימן השלם בשיטת המשלים ל-2. 0 עבור מספרים אי שלילים ו 1 עבור שליליים.

$$(5)_{10} = (\mathbf{0}0000000|00000000|00000000|00000101)_2 = (\mathbf{0}101)_2 \neq (\mathbf{1}01)_2$$

נתבונן כעת בייצוגים הבינארים המינימאליים של המספר 1 המספר 0 והמספר -1 בבסיס 10:

$$(1)_{10} = (\mathbf{0}1)_2 \neq (\mathbf{1})_2$$

$$(0)_{10} = (0)_2$$

$$(-1)_{10} = (\mathbf{1}1)_2 \neq (\mathbf{1})_2$$

ניתן לראות שגם בדוגמא זו כאשר אנו מנסים לייצג את -1 בבסיס בינארי מינימאלי כך ש n=1, אנו למעשה מאבדים את הייצוג הבינארי שלו בשיטת המשלים ל-2. וכאשר אנו מנסים לייצג את 1 עם n=1 אנחנו מייצגים ספרה ללא סימון. לכן נגדיר שהייצוג הבינארי $(\mathbf{1})_2$ אינו חוקי.

היוצא דופן הוא 0. מכיוון שהוא אינו מצריך סימן, ניתן להשתמש בו כייצוג בינארי המינימאלי של 0 בבסיס 10.

ייצוג נוסף שאינו חוקי הוא מספרים בינארים עם MSB שהוא 1 וכל שאר הביטים 0. לדוגמא 1000, 100000000,10. הייצוגים הללו אינם חוקיים מכיוון שאמנם הם מייצגים את המספר הקטן ביותר בייצוג המשלים ל-2 עבור n קבוע, לדוגמא עבור n=4, 1000 מייצג את -8. אך 1000 יכול להתקבל גם מחיבור של 01 עם 0111 (1+7) שמחזיר 1000 (-8) ונקבל פתרון לא נכון. לכן נגדיר שמספרים בינאריים מסוג זה אינם חוקיים עבור ייצוג בינארי מינימאלי. ובמקום עלינו להוסיף את הMSB המתאים על מנת לייצג את המספר בבסיס 10 המתאים. לדוגמא 01000 עבור 8 ו- 11000 עבור -8.

פרקטית, הדרך למצוא את הייצוג הבינארי המינימלי היא באמצאות העברת ה- MSB למקום שעדיין מקיים אותו ייצוג בבסיס 10.

$$(5)_{10} = (\mathbf{0}0000000|00000000|00000000|00000101)_2 = (\mathbf{0}0000101)_2 = (\mathbf{0}101)_2$$

$$(-5)_{10} = (\mathbf{1}1111111|11111111|11111111|11111011)_2 = (\mathbf{1}1111011)_2 = (\mathbf{1}011)_2$$

את אורך הייצוג בינארי מינימאלי עבור מספר שלם i שונה מ-0 ניתן לחשב על ידי הנוסחא:

$$log_2(|i|) + 2$$

את התוצאה שנקבל נעגל כלפי מטה על מנת לקבל את האורך הרצוי, לדוגמא:

$$log_2(|5|) + 2 = 4.321 \ \rightarrow 4$$

בטבלה הבאה ניתן לראות דוגמא לשימוש בייצוג בינארי בשיטת המשלים ל-2 לצד הייצוג הבינארי המינימאלי, עבור n=4.

המספר	ייצוג ב 4 ביטים	ייצוג בינארי מינימאלי
7	*0111*	*0111*
6	*0110*	*0110*
5	*0101*	*0101*
4	*0100*	*0100*
3	*0011*	*011*
2	*0010*	*010*
1	*0001*	*01*
0	*0000*	*0*
-1	*1111*	*11*
-2	*1110*	*110*
-3	*1101*	*101*
-4	*1100*	*1100*
-5	*1011*	*1011*
-6	*1010*	*1010*
-7	*1001*	*1001*
-8	*1000*	*11000*

<u>שיטת המשלים ל – 2</u>

שיטת המשלים ל־2 היא שיטה לייצוג מספרים עם סימן בבסיס בינארי . בשיטה זאת הסיבית הגבוהה ביותר (MSB - Most Significant Bit) מייצגת את הסימן של המספר (חיובי או שלילי) ושאר הספרות מייצגות את ערך המספר (בצורה שונה מייצוג רגיל אם הוא שלילי). שיטה זו מקובלת בתחום המחשבים כשיטה שימושית לייצוג בינארי של מספרים שעשויים להיות שליליים או חיוביים. השימוש במשלימים במחשבים נועד לפשט את פעולת החיסור, וכן לביצוע פעולות לוגיות.

השיטה מתבצעת באופן הבא, לדוגמא אם ברשותי 10 ערכים בינאריים כלשהם $x_1 \ , x_2, \dots x_8, x_9, x_{10}$. באופן טבעי נבחר לייצגם עם ערכים בבסיס 10 באופן הבא:

$$x_1 \ \rightarrow 0 \, , x_2 \rightarrow 1, \dots x_8 \rightarrow 7, x_9 \rightarrow 8, x_{10} \rightarrow 9$$

אך במידה ואנו מעוניינים להשתמש בייצוגים שליליים נאלץ להקצות חצי עבור ערכים שלילים באותו האופן:

$x_1 \ \rightarrow 0$	$x_2 \ \rightarrow 1$	$x_3 \ \rightarrow 2$
$x_4 \ \rightarrow 3$	$x_5 \ \rightarrow 4$	$x_6 \ \rightarrow -5$
$x_7 \ \rightarrow -4$	$x_8 \ \rightarrow -3$	$x_9 \ \rightarrow -2$
$x_{10} \ \rightarrow -1$		

הדרך לעשות זאת עם מספרים בינאריים היא על ידי שימוש ב- MSB, כי חצי מהמספרים הבינאריים הם עם MSB 1 וחצי עם MSB 0.

על מנת למצוא את הייצוג הבינארי השלילי של מספר כלשהו ניתן לבצע את הצעדים הבאים:

- הפיכת כל הביטים .
- ביצוע פעולת חיבור עם 1.

לדוגמא: נעבור מ- 5 ל- 5-.

$$(5)_{10} = \ (0101)_2 \quad \xrightarrow{\text{הפיכת כל הביטים}} \quad (1010)_2 \quad \xrightarrow{\text{פעולת חיבור עם 1}} \quad + \begin{array}{r} 1010 \\ 1 \\ \hline 1011 \end{array} \quad \longrightarrow \quad (1011)_2 = \ (-5)_{10}$$

באופן יותר פורמאלי, בהיתנן מספר חיובי N לפי בסיס 2, הכולל n ספרות של ביטים, המשלים ל-2 של N מוגדר כ-:

$$f(N,n) = \begin{cases} 2^n - N & if\ N \neq 0 \\ 0 & if\ N = 0 \end{cases}$$

לדוגמא:

$N = (0101)_2 = (5)_{10}$

$n = 10$

$\Rightarrow$

$N = (0000000101)_2$

$$f\left((0000000101)_2, 10\right) = (2^{10})_{10} - (0000000101)_2$$
$$= (10000000000 - 0000000101)_2$$
$$= (1111111011)_2$$
$$= (-5)_{10}$$

אריתמטיקה

פעולות אריתמטיות של מספרים לפי בסיס 2 נעשות לפי אותם הכללים הנהוגים בפעולות המספרים בבסיס 10. כאשר מחשבים לפי בסיס 2, יש להקפיד ולהשתמש אך ורק ב-2 הספרות המותרות לשימוש, כלומר 1 ו – 0.

להלן דוגמאות לחיבור שני מספרים בינאריים מינימאליים:

-19 + (-25)

$$\begin{array}{r} 101101 \\ +\ \ 100111 \\ \hline 1010100 \end{array}$$

15 + 1

$$\begin{array}{r} 01111 \\ +\ \ 01 \\ \hline 010000 \end{array}$$

פעולת החיסור בשיטת המשלים ל-2 של מספרים בינאריים מינימאלים היא מעט שונה מן השיטות המוכרות. אפשר לחסר שני מספרים חיוביים (M-N) לפי בסיס 2 כדלהלן:

- בצעו פעולת חיבור בין M למשלים ל-2 של N. אם מספר הביטים של M ו N אינו שווה רפדו בעזרת ה-MSB את הביטוי הבינארי עם מספר הביטים המועט יותר.
- לאחר חישוב השלב הראשון:
  - אם יש נשא סופי - התעלמו ממנו, והחזירו את התוצאה.
  - אם אין נשא סופי - החזירו את התוצאה.

יש לציין שהתוצאה אינה תמיד מתקבלת כיצוג בינארי מינימלי.

84 - 68 $\Rightarrow$ $\begin{array}{r} 01010100 \\ -\ \ 01000100 \\ \hline \end{array}$ $\Rightarrow$ $\begin{array}{r} 01010100 \\ +\ \ 10111100 \\ \hline 100010000 \end{array}$ $\Rightarrow$ 00010000

נשא סופי

68 - 84 $\Rightarrow$ $\begin{array}{r} 01000100 \\ -\ \ 01010100 \\ \hline \end{array}$ $\Rightarrow$ $\begin{array}{r} 01000100 \\ +\ \ 10101100 \\ \hline 11110000 \end{array}$ $\Rightarrow$ 11110000

אין נשא סופי

3 - 68 $\Rightarrow$ $\begin{array}{r} 00000011 \\ -\ \ 01000100 \\ \hline \end{array}$ $\Rightarrow$ $\begin{array}{r} 00000011 \\ +\ \ 10111100 \\ \hline 10111111 \end{array}$ $\Rightarrow$ 10111111

אין נשא סופי

-3 - 68 $\Rightarrow$ $\begin{array}{r} 11111101 \\ -\ \ 01000100 \\ \hline \end{array}$ $\Rightarrow$ $\begin{array}{r} 11111101 \\ +\ \ 10111100 \\ \hline 110111001 \end{array}$ $\Rightarrow$ 10111001

נשא סופי

בפעולת כפל וחילוק ניתן להשתמש בכפל/ חילוק ארוך כפי שנלמד בבתי הספר היסודיים אך אם נתבונן בפעולת הכפל, באופן מופשט ניתן לפתור בעיית כפל על ידי חיבור חוזר:

$$\boxed{n * m = \underbrace{(n + n + \ldots\ldots + n)}_{\text{m times}}}$$

ובאופן דומה ניתן לפתור בעיית חילוק. אך הפיתרונות אינם יעילים, אבל בהחלט ניתן לייעל את הפתרון המופשט על ידי כלים שנלמדו בקורס. כלומר במקום שנבצע פעולות חיבור אחד אחרי השני בלולאה, ניתן לפצל את פעולת החיבור ולבצעם במקביל.

$$n * m = \underbrace{(n + n + \ldots\ldots + n)}_{\text{m times}}$$

$$n * \frac{1}{2}m = \underbrace{(n + n + \ldots\ldots + n)}_{\frac{1}{2}\text{m times}} \quad + \quad n * \frac{1}{2}m = \underbrace{(n + n + \ldots\ldots + n)}_{\frac{1}{2}\text{m times}}$$

ובעבור חילוק $\frac{n}{m}$ נוכל להגדיל את המכנה או להקטין את המונה.

1) הקטנת המונה n בחצי עד מקרה בסיס שבו הביטוי $\frac{n}{m}$ שווה ל 1 או קטן מ-1 ולבסוף סכימת מקרי בסיס מתאימים:

$$\boxed{\frac{n}{m}}$$

$$\boxed{\frac{0.5 * n}{m}} \quad + \quad \boxed{\frac{0.5 * n}{m}}$$

שימו לב שאם בחרתם לממש שיטה זו בעזרת השיטה *divideBy2()* הנתונה לכם, תוצאת החילוק מחזירה תמיד מספר שלם. לכן עליכם למצוא דרך להתחשב בשאריות לאורך החלוקות.

שימו לב שבמימוש *divide()* בעבודה אתם נדרשים להחזיר מספר שלם.

2) הגדלת המכנה m פי 2 עד מקרה שבו הביטוי $\frac{n}{m}$ קטן מ-1. ולאחר מכן ביצוע פעולות אריטמתיות מתאימות עם מספר הפעמים שכפלנו את m כדי להחזיר תוצאה נכונה.

$$\boxed{\frac{n}{m}}$$

$$\downarrow$$

$$\boxed{\frac{n}{2*m}}$$

(נסו זאת בעצמכם עם מספרים בבסיס 10)