W komputerach używa się zwykle do zapisu liczby 8, 16, 32 lub 64 miejsc. Przy t miejscach możemy zapisać 2^t różnych liczb.

Komputery i liczby

Mgr Teresa PRZYTYCKA

3 2 1 0

| 0 | 0 | 0 | 0 | = 0

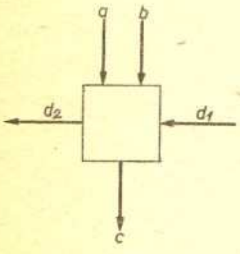
⋮

| 0 | 1 | 1 | 0 | = 6

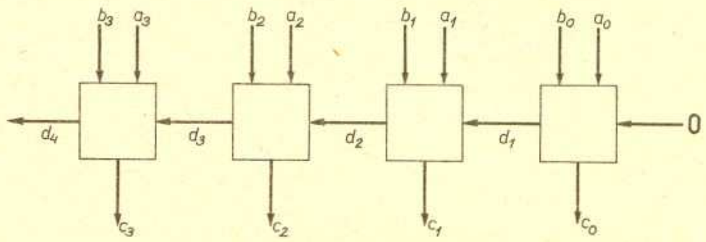
⋮

| 1 | 1 | 1 | 1 | = 15

Wszystkim zapewne wiadomo, że komputery rachują na liczbach zapisanych w układzie dwójkowym. Jak jednak zakodować liczbę, by rachunki na niej były łatwe? Przypuśćmy, że zapisujemy liczbę w 4 komórkach, w każdej zaś komórce możemy zapisać 0 lub 1. Jeśli ograniczymy się do liczb nieujemnych, to korzystając ze zwykłego zapisu dwójkowego kodujemy liczby od 0 do $2^4-1 = 15$. Takie liczby możemy dodawać budując z układów elementarnych (rys. 1) sumator (rys. 2). Sumator ten dodaje liczby $a_3a_2a_1a_0$ i $b_3b_2b_1b_0$, otrzymując $c_3c_2c_1c_0$.



Rys. 1. Układ elementarny dodaje 3 liczby a, b, d_1 równe 0 lub 1 i otrzymany wynik podaje w postaci dwóch cyfr d_2 i c, tzn. $a+b+d_1 = 2d_2+c$.



0110 + 0101 = 1011   i $d_4=0$
6 + 5 = 11

1010 + 0111 = 0001   i $d_4=1$
10 + 7 ≠ 1

Rys. 2

Czasem jednak otrzymujemy zły wynik. Następuje to wtedy, gdy suma przekracza 15. Stan ten nazywany jest przepełnieniem. Ale wtedy $d_4 = 1$ i w ten sposób przepełnienie można wykryć.

Teraz chcemy kodować również liczby ujemne. Najprostszy sposób to zapisanie znaku liczby na jednym z miejsc, np. miejscu 3, a na pozostałych trzech wartości bezwzględnej liczby:

$$r_1(x) = \begin{cases} 0x_B & \text{dla } x \geqslant 0 \\ 1|x|_B & \text{dla } x \leqslant 0, \end{cases}$$

gdzie x_B oznacza zapis dwójkowy liczby nieujemnej x. Możemy tak zapisywać liczby od -7 do $+7$.

$r_1(5)$ = 0101 = 5

$r_1(-3)$ = 1011 = −3

$r_1(0)$ = 1000 lub 0000

0100 + 0010 = 0110
4 + 2 = 6

1011 + 0100 = 1111
−3 + 4 ≠ −7

Niestety, dodając tak zapisane liczby sumatorem z rysunku 2 możemy otrzymać bezsensowny wynik. (Dodawanie liczb dodatnich o sumie mniejszej niż 8 wykonywane jest nadal poprawnie.)
Zakodujmy liczby inaczej

$$r_2(x) = \begin{cases} 0x_B & \text{dla } x \geqslant 0 \\ 1\overline{|x|}_B & \text{dla } x \leqslant 0, \end{cases}$$

gdzie $\overline{y_B}$ oznacza zmianę wszystkich cyfr na inne (tzn. 0 na 1 i 1 na 0) i spróbujmy je dodawać na naszym sumatorze.

$r_2(-3)$ = 1100

$r_2(-7)$ = 1000

$r_2(0)$ = 1111 lub 0000

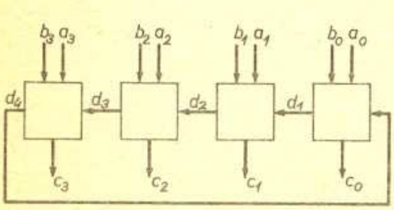
0011 + 0010 = 0101
3 + 2 = 5

1100 + 0010 = 1110
−3 + 2 = −1

0011 + 1101 = 0000   i $d_4=1$
3 + −2 ≠ 0

1100 + 1101 = 1001   i $d_4=1$
−3 + −2 ≠ −6

Zauważmy, że błędne wyniki są o 1 za małe i wtedy $d_4 = 1$. Nie jest to przypadek. Jest tak zawsze, gdy obie liczby są ujemne lub gdy ujemna ma mniejszą wartość bezwzględną. Można zatem nasz sumator poprawić. Najpierw dodaje on liczby a i b, następnie do wyniku dodaje liczbę d_4 (0 lub 1). Sprawdź Czytelniku, że tak poprawiony sumator dodaje prawidłowo (dowód w numerze). Przy tym sposobie kodowania przepełnienie wykrywamy badając, czy $d_4 = d_3$.



Kodowanie r_2, podobnie jak r_1, ma wadę. 0 ma dwa zapisy. Okazuje się, że na te „dwa 0" też jest lekarstwo. Kod r_3 nie ma tej wady:

$$r_3(x) = \begin{cases} 0x_B & \text{dla } x \geqslant 0 \\ 1\overline{|x|}_B+1 & \text{dla } x \leqslant 0. \end{cases}$$

Liczby zapisane tym kodem możemy dodawać pierwszym z sumatorów otrzymując prawidłowe wyniki. Kodem r_3 można zapisać liczby od -8 do 7.