System binarny jest opisany tylko na liczbach 0 i 1 i 1-dnym bicie

System 8-kowy jest opisany na liczbach [0, 1, 2, 3, 4, 5, 6, 7] i opisujemy go na 3 bitach

System 16-kowy jest opisany na liczbach [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F] i opisany jest na 4-rech bitach dodatkowo trzeba pamiętać że cyfry: A = 10, B = 11, C = 12, D = 13, E = 14, F = 15

Warto zapamiętać **rozwinięcie potęgowe liczby 2 na 8 bitach** ☺

128, 64, 32, 16, 8, 4, 2, 1

Pokażę w jak łatwy sposób można zamienić systemy liczb nie korzystając z dzielenia tylko pomocy systemu binarnego.

Zamiana z systemu dziesiętnego$_{(10)}$ na binarny$_{(2)}$. Bierzemy liczbę i patrzymy na rozwinięcie potęgowe liczby 2 staramy się dobierać najwyższą możliwą liczbę np.

- $103_{(10)}$

128	64	32	16	8	4	2	1
0	1	1	0	0	1	1	1

Wynik: $01100111_{(2)}$

- $24_{(10)}$

128	64	32	16	8	4	2	1
0	0	0	1	1	0	0	0

Wynik: $00011000_{(2)}$

- $255_{(10)}$

128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1

Wynik: $11111111_{(2)}$

Dzięki zamianie z dziesiętnego na binarny oraz odwrotnie polegającego na tej samej zasadzie patrzenia na rozwinięcie potęgowe liczby 2 np. (wystarczy że będziemy dodawać miejsca gdzie są 1)

- $00000000_{(2)}$

128	64	32	16	8	4	2	1
0	0	0	0	0	0	0	0

Wynik: $0_{(10)}$

- $10101010_{(2)}$

128	64	32	16	8	4	2	1
1	0	1	0	1	0	1	0

Wynik: $170_{(10)}$

- $01101110_{(2)}$

128	64	32	16	8	4	2	1
0	1	1	0	1	1	1	0

Wynik: $110_{(10)}$

Znając system binarny możemy praktycznie zamieniać na każdy inny system tylko wystarczy zapamiętać iż system 8-kowy opisany jest na 3 bitach a 16 jest opisany na 4 bitach

Jak prosto zamienić liczbę dziesiętną na system 8-kowy lub 16-kowy? Wykorzystam do tego system binarny ze względu na to iż bardzo łatwo jest zamienić z dziesiętnego na binarny a mając już tzw. notację 0, 1 mogę podzielić na odpowiednie bity i zapisać odpowiednimi literami ☺ na koniec dodam tabelkę która pokazuje każdą cyfrę w systemie 0 i 1.

- $103_{(10)}$
  Najpierw zamieniam na system binarny na 8 bitach.

128	64	32	16	8	4	2	1
0	1	1	0	0	1	1	1

Mając tak zamienioną liczbę mogę teraz podzielić na odpowiednie bity dla 16 są to 4 a dla 8 są to 3.

Zacznę od 16-kowego ze względu na to że łatwiej podzielić.

8	4	2	1	8	4	2	1
0	1	1	0	0	1	1	1
6				7			

Wynik: $67_{(16)}$
(Dlaczego zamieniłem w tabelce potęgi 2? Ze względu na to, że opisujemy to tylko na 4 bitach maksymalna liczba w 16 to F co daje nam 15, a dodanie do siebie 1+2+4+8 = 15 ☺ to samo dzieje się w 8-kowym tylko że tam maksymalna liczba to 7 dlatego 3 bity)

Teraz zamienię na system 8-kowy.

2	1	4	2	1	4	2	1
0	1	1	0	0	1	1	1

Ze względu na to iż 8-kowy można opisać na 3 bitach muszę dopisać brakujące bity z przodu tablicy (to samo można zrobić z systemem 16-kowym jeśli brakuje)

4	2	1	4	2	1	4	2	1
0	0	1	1	0	0	1	1	1

Teraz mogę podzielić na równe części i obliczyć:

4	2	1	4	2	1	4	2	1
0	0	1	1	0	0	1	1	1
1			4			7		

Wynik: $147_{(8)}$

Działa to też w drugą stronę kiedy chcemy zamienić z innego systemu na system dziesiętny:

- $F1_{(16)}$
  Rozpisujemy F na 4bitach oraz 1 na kolejnych 4bitach

8	4	2	1	8	4	2	1
F = 15				1			
1	1	1	1	0	0	0	1

Mając liczbę w systemie binarnym $11110001_{(2)}$ już łatwo potrafimy zamienić ją na 10-siętny.

128	64	32	16	8	4	2	1
1	1	1	1	0	0	0	1

Wynik: $241_{(10)}$

- $377_{(8)}$
  Rozpisuję każdą liczbę osobno na 3 bitach

4	2	1	4	2	1	4	2	1
3			7			7		
0	1	1	1	1	1	1	1	1

Posiadając liczbę w systemie binarnym $011111111_{(2)}$ możemy pierwsze 0 usunąć ze względu na to że nie zmienia nic w liczbie patrzymy tylko na 1-ki.

128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1

Wynik: $255_{(10)}$

BINARNY				ÓSEMKOWY	DZIESIĘTNY	SZESTNASTWOY
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	2	2	2
0	0	1	1	3	3	3
0	1	0	0	4	4	4
0	1	0	1	5	5	5
0	1	1	0	6	6	6
0	1	1	1	7	7	7
1	0	0	0		8	8
1	0	0	1		9	9
1	0	1	0			A = 10
1	0	1	1			B = 11
1	1	0	0			C = 12
1	1	0	1			D = 13
1	1	1	0			E = 14
1	1	1	1			F = 15

Systemy binarne z zapisem liczb ujemnych.

System ZM – Znak Moduł dla mnie najprostszy system pisania liczb ujemnych. Pierwszy bit pokazuje nam czy jest to liczba ujemna (1) czy dodatnia (0)

Przykłady zamiany liczby binarnej na dziesiętną:

- $10011010_{(ZM)}$

  Rozpisujemy liczbę:

ZM	64	32	16	8	4	2	1
1	0	0	1	1	0	1	0

  ZM = 1 czyli jest to liczba ujemna

  16+8+2 = 26

  Wynik: $-26_{(10)}$

- $011010_{(ZM)}$

  Rozpisujemy liczbę:

ZM	16	8	4	2	1
0	1	1	0	1	0

  ZM = 0 jest to liczba dodatnia

  16+8+2 = 26

  Wynik: $26_{(10)}$

- $110101010_{(ZM)}$

  Rozpisujemy liczbę:

ZM	128	65	32	16	8	4	2	1
1	1	0	1	0	1	0	1	0

  ZM = 1 liczba ujemna

  Wynik: $-170_{(10)}$

- $010101010_{(ZM)}$

  Rozpisujemy liczbę:

ZM	128	65	32	16	8	4	2	1
0	1	0	1	0	1	0	1	0

  ZM = 0 liczba dodatnia

  Wynik: $170_{(10)}$

Zamiana z liczny dziesiętnej na liczbę ZM

- $152_{(10)}$

  Rozpisujemy liczbę binarnie

128	64	32	16	8	4	2	1
1	0	0	1	1	0	0	0

  Dopisujemy na początku ZM oraz usuwany niepotrzebne zera

  Wynik: $010011000_{(ZM)}$

- $-152_{(10)}$

  Rozpisujemy liczbę binarnie

128	64	32	16	8	4	2	1
1	0	0	1	1	0	0	0

  Dopisujemy na początku ZM oraz usuwany niepotrzebne zera

  Wynik: $110011000_{(ZM)}$

- $33_{(10)}$

  Rozpisujemy liczbę binarnie

128	64	32	16	8	4	2	1
0	0	1	0	0	0	0	1

  Dopisujemy na początku ZM oraz usuwany niepotrzebne zera

ZM	128	64	32	16	8	4	2	1
0	0	0	1	0	0	0	0	1

ZM	32	16	8	4	2	1
0	1	0	0	0	0	1

  Wynik: $0100001_{(ZM)}$

- $-33_{(10)}$

  Rozpisujemy liczbę binarnie

128	64	32	16	8	4	2	1
0	0	1	0	0	0	0	1

  Dopisujemy na początku ZM oraz usuwany niepotrzebne zera

ZM	128	64	32	16	8	4	2	1
1	0	0	1	0	0	0	0	1

ZM	32	16	8	4	2	1
1	1	0	0	0	0	1

  Wynik: $1100001_{(ZM)}$

System U1 tak samo jak w ZM pierwszy bit oznacza czy jest to liczba ujemna (1) czy dodatnia (0) z tym że przy liczbach ujemnych liczba binarna jest negowana.

Zamiana z liczby dziesiętnej na U1

- 166

  Zapisujemy liczbę w systemie binarnym

128	64	32	16	8	4	2	1
1	0	1	0	0	1	1	0

  Ma być to liczna dodatnia czyli nie negujemy dopisujemy z przodu tylko znak U1 i otrzymujemy wynik

U1	128	64	32	16	8	4	2	1
0	1	0	1	0	0	1	1	0

  Wynik: $010100110_{(U1)}$

- -166

  Zapisujemy liczbę w systemie binarnym

128	64	32	16	8	4	2	1
1	0	1	0	0	1	1	0

  Ma być to liczna ujemna czyli negujemy dopisujemy z przodu tylko znak U1 i otrzymujemy wynik

  Negacja:

128	64	32	16	8	4	2	1
1	0	1	0	0	1	1	0
0	1	0	1	1	0	0	1

  Dopisanie U1

U1	128	64	32	16	8	4	2	1
1	0	1	0	1	1	0	0	1

  Wynik: $101011001_{(U1)}$

- 21

  Zapisujemy liczbę w systemie binarnym

128	64	32	16	8	4	2	1
0	0	0	1	0	1	0	1

  Ma być to liczna dodatnia czyli nie negujemy dopisujemy z przodu tylko znak U1 i otrzymujemy wynik

U1	128	64	32	16	8	4	2	1
0	0	0	0	1	0	1	0	1

U1	16	8	4	2	1
0	1	0	1	0	1

  Wynik: $010101_{(U1)}$
- -21

  Zapisujemy liczbę w systemie binarnym

128	64	32	16	8	4	2	1
0	0	0	1	0	1	0	1

  Ma być to liczna ujemna czyli negujemy dopisujemy z przodu tylko znak U1 i otrzymujemy wynik

  Usunięcie niepotrzebnych zer

16	8	4	2	1
1	0	1	0	1

  Negacja:

16	8	4	2	1
1	0	1	0	1
0	1	0	1	0

  Dopisanie U1

U1	16	8	4	2	1
1	0	1	0	1	0

  Wynik: $101010_{(U1)}$

Zamiana z liczby U1 na dziesiętną

- 01010101

  Rozpisujemy liczbę

U1	64	32	16	8	4	2	1
0	1	0	1	0	1	0	1

  U1 = 0 jest to liczba dodatnia nie jest negowana

  Wynik: $85_{(10)}$

- 11010101

  Rozpisujemy liczbę

U1	64	32	16	8	4	2	1
1	1	0	1	0	1	0	1

  U1 = 1 jest to liczba ujemna musimy zanegować aby otrzymać wynik

64	32	16	8	4	2	1
1	0	1	0	1	0	1
0	1	0	1	0	1	0

  Wynik: $-42_{(10)}$

- 010011001

  Rozpisujemy liczbę

U1	128	64	32	16	8	4	2	1
0	1	0	0	1	1	0	0	1

  U1 = 0 jest to liczba dodatnia nie jest negowana

  Wynik: $153_{(10)}$

- 110011001

  Rozpisujemy liczbę

U1	128	64	32	16	8	4	2	1
1	1	0	0	1	1	0	0	1

  U1 = 1 jest to liczba ujemna musimy zanegować aby otrzymać wynik

128	64	32	16	8	4	2	1
1	0	0	1	1	0	0	1
0	1	1	0	0	1	1	0

  Wynik: $-102_{(10)}$

System U2 nie wiem jak wyjaśnić dobrze ten system jest dla mnie ułomny. nadaje się do liczb ujemnych i w odróżnieniu od U1 tutaj zamiast negować odejmujemy. Są dwa sposoby obliczania pokażę ten który uważam za łatwiejszy.

Na system U2 z dziesiętnego

- $166_{(10)}$

  Akurat tej liczby na 8 bitach nie możemy zapisać ale całość wygląda tak jak normalnie napisanie liczby binarnej dopisanie na początku 0 bo jest to liczba dodatnia, usunięcie nadmiarowych zer o ile występują i otrzymujemy wynik.

128	64	32	16	8	4	2	1
1	0	1	0	0	1	1	0

U2	128	64	32	16	8	4	2	1
0	1	0	1	0	0	1	1	0

Wynik: $010100110_{(U2)}$

- $-166_{(10)}$
  Przy ujemnych liczbach musimy wiedzieć jaką wartość ma U2 ponieważ od tej wartości odejmujemy liczbę a później zapisujemy. Widzimy że liczba 166 jest większa od 128 więc bit znaczący będzie mieć wartość 256 oraz będziemy opisywać na 9 bitach

  Musimy teraz obliczyć wartość kodu pozostałych bitów
  256 – 166 = 90

  Posiadając liczbę 90 przerabiamy na postać binarną dopisujemy na początku znak U2 i posiadamy już wynik

U2	128	64	32	16	8	4	2	1
1	0	1	0	1	1	0	1	0

Wynik: $101011010_{(U2)}$

- $21_{(10)}$

128	64	32	16	8	4	2	1
0	0	0	1	0	1	0	1

U2	128	64	32	16	8	4	2	1
0	0	0	0	1	0	1	0	1

U2	16	8	4	2	1
0	1	0	1	0	1

Wynik: $010101_{(U2)}$

- $-21_{(10)}$
  Tą liczbę możemy zrobić na 8 bitach tak więc bit znaczący będzie miał wartość 128. Całość wygląda następująco:

  128 – 21 = 107

U2	64	32	16	8	4	2	1
1	1	1	0	1	0	1	1

Wynik: $11101011_{(U2)}$

Na dziesiętny z U2

- $01010101_{(U2)}$
  Tutaj mamy zwykłe przeliczenie z binarnego na dziesiętny

U2	64	32	16	8	4	2	1
0	1	0	1	0	1	0	1

Wynik: 85

- $11010101_{(U2)}$
  Z przodu stoi jedynak jest to znak U2 czyli liczba jest ujemna mamy 8 bitów tak więc bit znaczący posiada wartość 128 teraz wystarczy obliczyć resztę wartości

U2	64	32	16	8	4	2	1
1	1	0	1	0	1	0	1

64+16+4+1 = 85

Posiadając obie wartości odejmujemy je

128-85=43

Wiemy że ma być to liczba ujemna tak więc wynik będzie:
Wynik: -43

- $010011001_{(U2)}$

U2	128	64	32	16	8	4	2	1
0	1	0	0	1	1	0	0	1

Wynik: 153

- $110011001_{(U2)}$
  Wartość jest opisana na 9 bitach czyli bit znaczący będzie miał wartość 256 zostaje obliczyć resztę:

U2	128	64	32	16	8	4	2	1
1	1	0	0	1	1	0	0	1

128+16+8+1 = 153

256 – 153 = 103

Wynik: -103