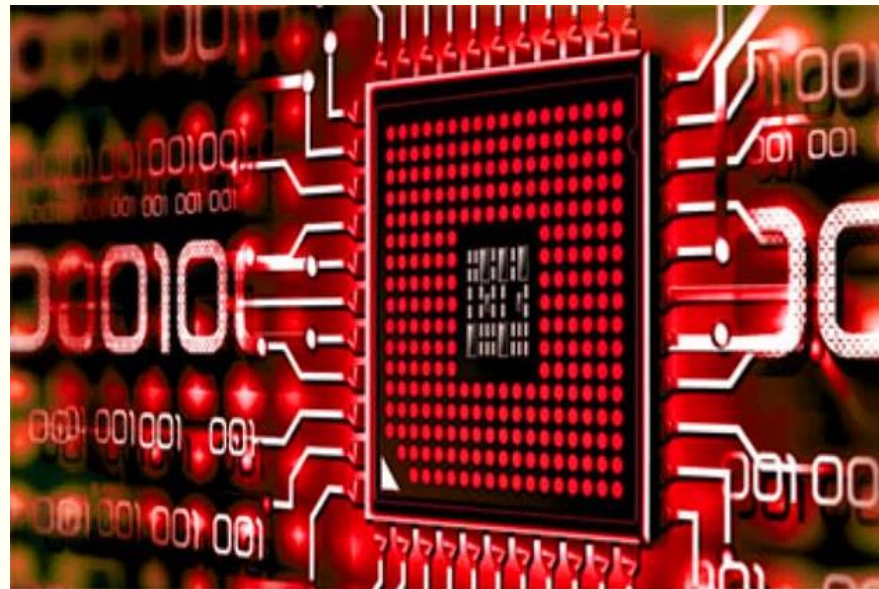




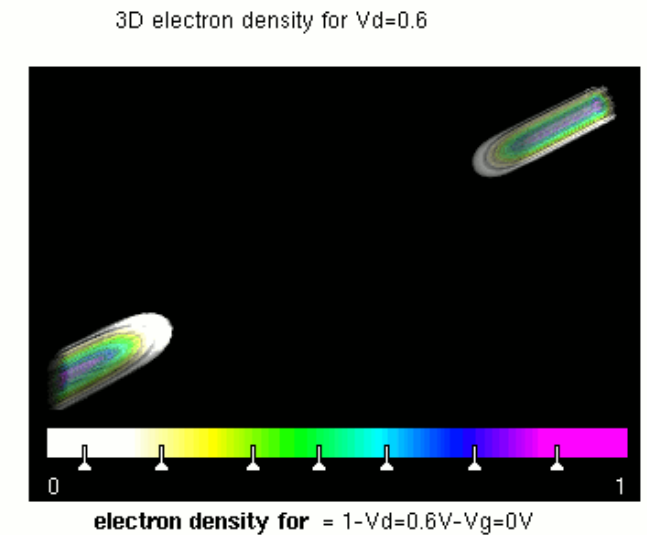
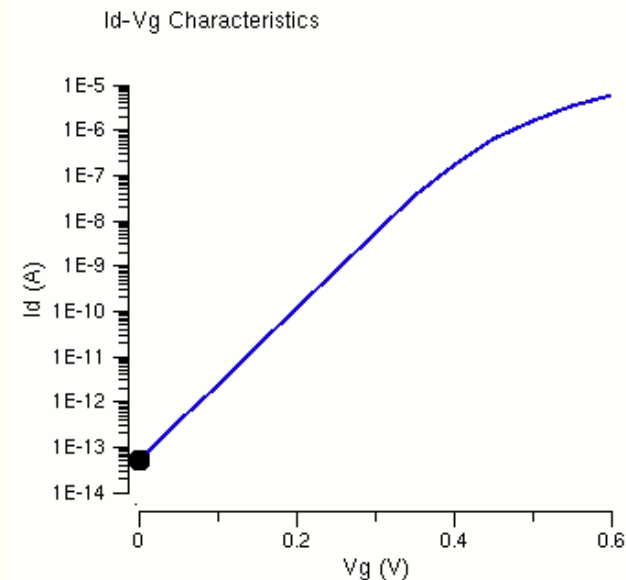
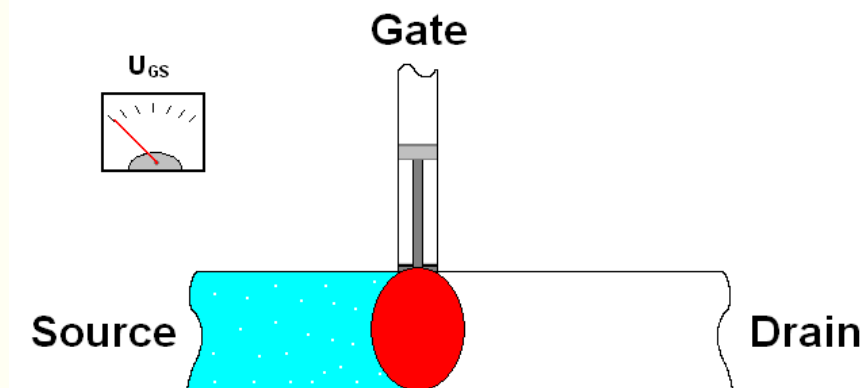
ЛОГІКА РОБОТИ ЦИФРОВИХ СИСТЕМ



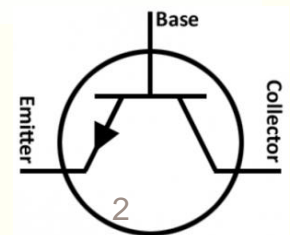
Сучасна техніка будується на транзисторах



- Напівпровідникові пристрої призначені для управління електричним струмом.
 - Можуть працювати як перемикачі
 - Здатні посилювати сигнал

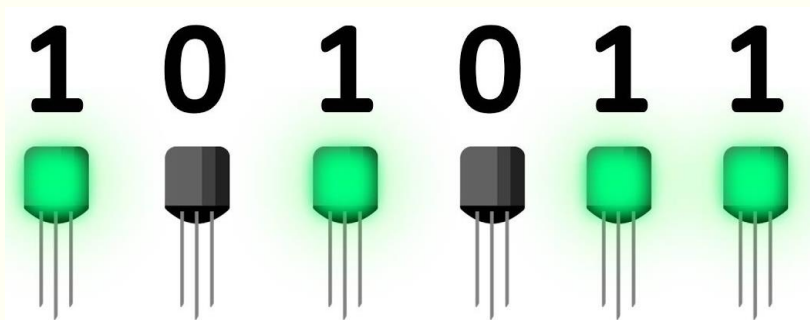


Тема доповіді: фізичні принципи роботи сучасних транзисторів

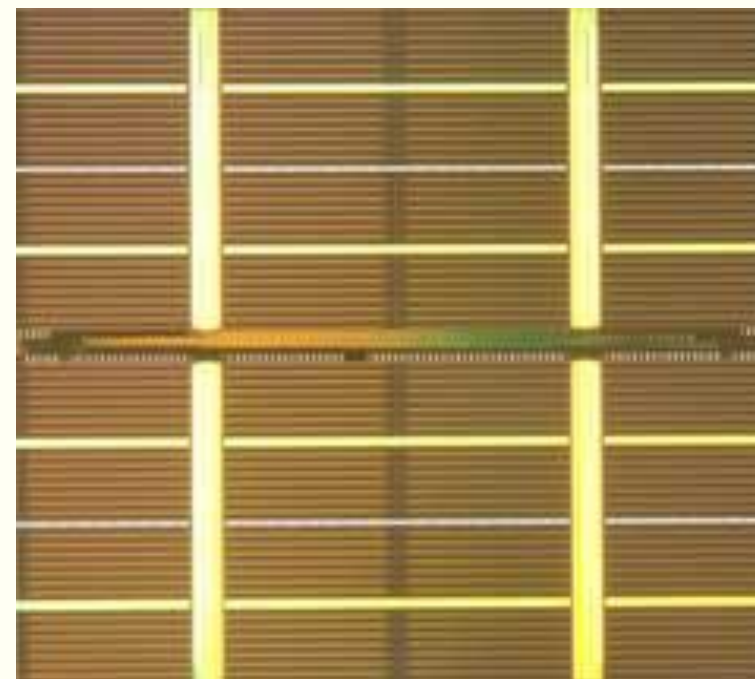


Мова комп'ютера – 0 і 1

- «**binary digit**» = біт (0 – струму немає, 1 – струм є)



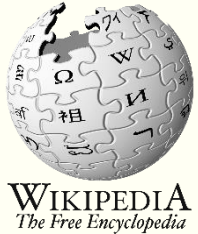
- Пам'ять комп'ютера – сітка комірок (memory cells), які, в тому числі, включають транзистори.
- Вся інформація кодується нулями та одиницями!
 - Числа
 - Текст
 - Зображення
 - Аудіо
 - Відео



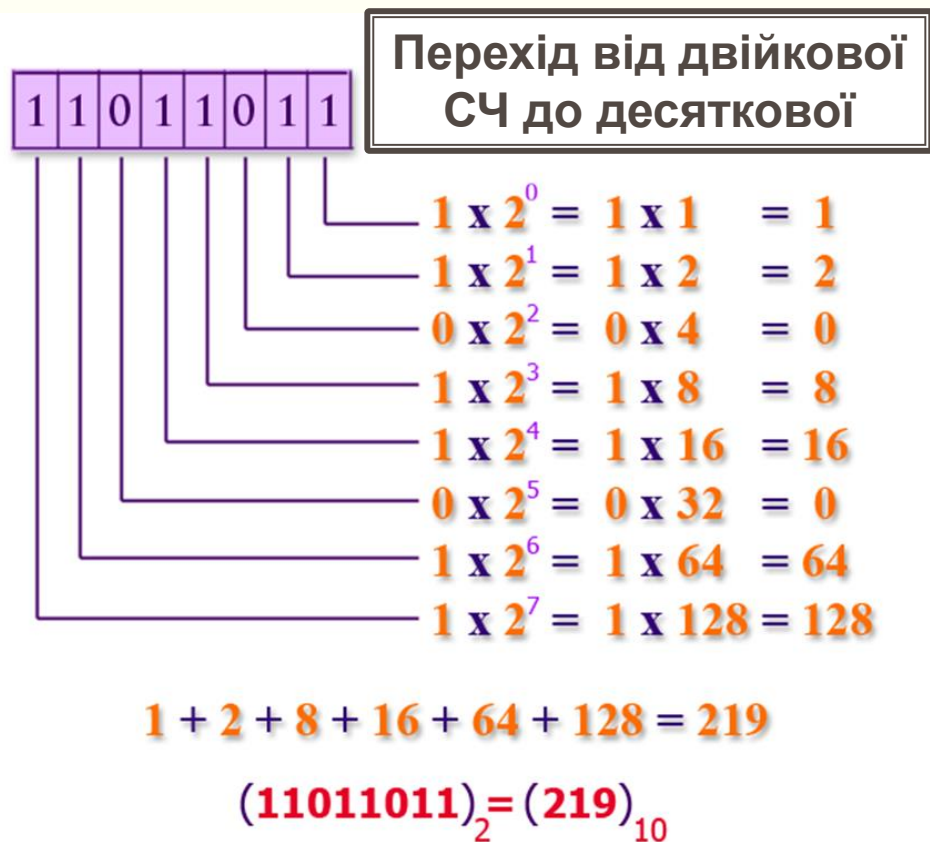
Системи числення

10	2	8	16
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

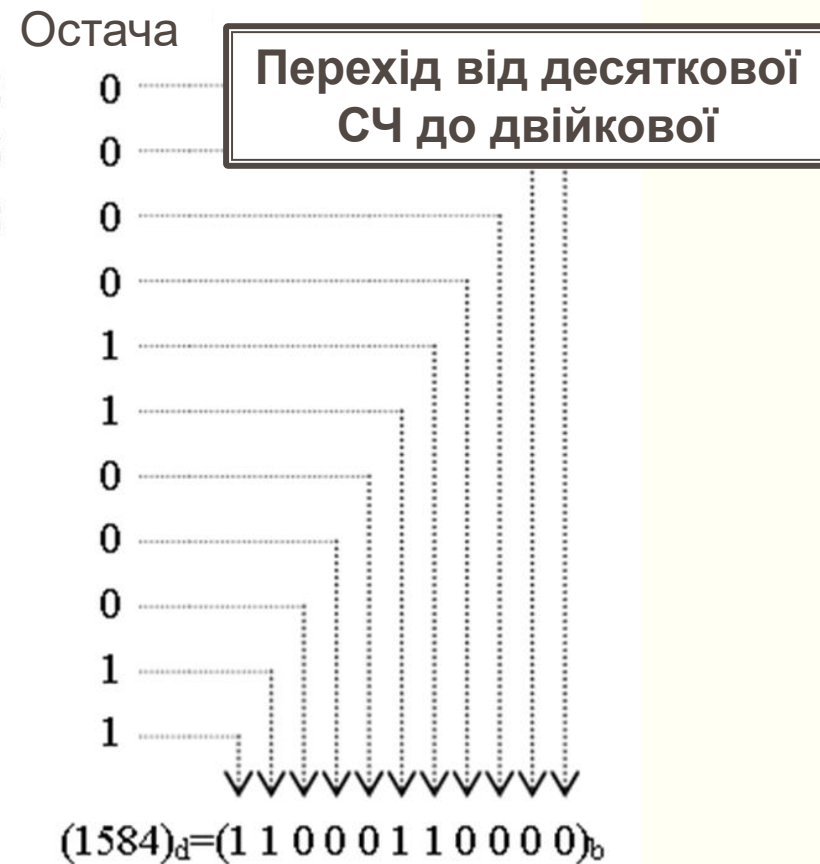
- **Системою числення**, або нумерацією, називається сукупність правил і знаків, за допомогою яких можна відобразити (кодувати) будь-яке невід'ємне число.
- За кількістю унікальних символів для представлення числа:
 - Двійкова (0, 1)
 - Вісімкова (0, 1, 2, 3, 4, 5, 6, 7)
 - Десяткова (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
 - Шістнадцяткова (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)



Двійкова система числення



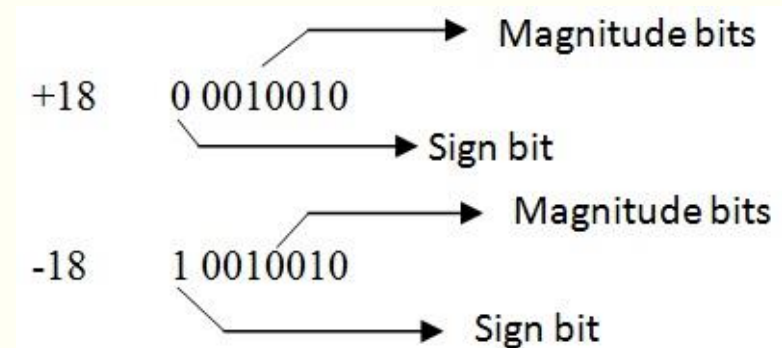
1584	:	2	=	792
792	:	2	=	396
396	:	2	=	198
198	:	2	=	99
99	:	2	=	49
49	:	2	=	24
24	:	2	=	12
12	:	2	=	6
6	:	2	=	3
3	:	2	=	1
1	:	2	=	0



Представлення даних. Цілі числа

- 1 байт = 8 послідовних бітів
 - 8-бітове ціле беззнакове число може представити діапазон від 0 до $2^8-1=255$
- 8-бітове ціле знакове число виділяє старший біт для знаку
 - діапазон від $-2^7=-128$ до $2^7-1=127$
- Байт – найменша комірка пам'яті, яку можна адресувати.
 - Комірки пам'яті пронумеровані, порядковий номер комірки – це її адреса.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
0	0	1	1	0	0	0	1	
0	0	32	16	0	0	0	1	49

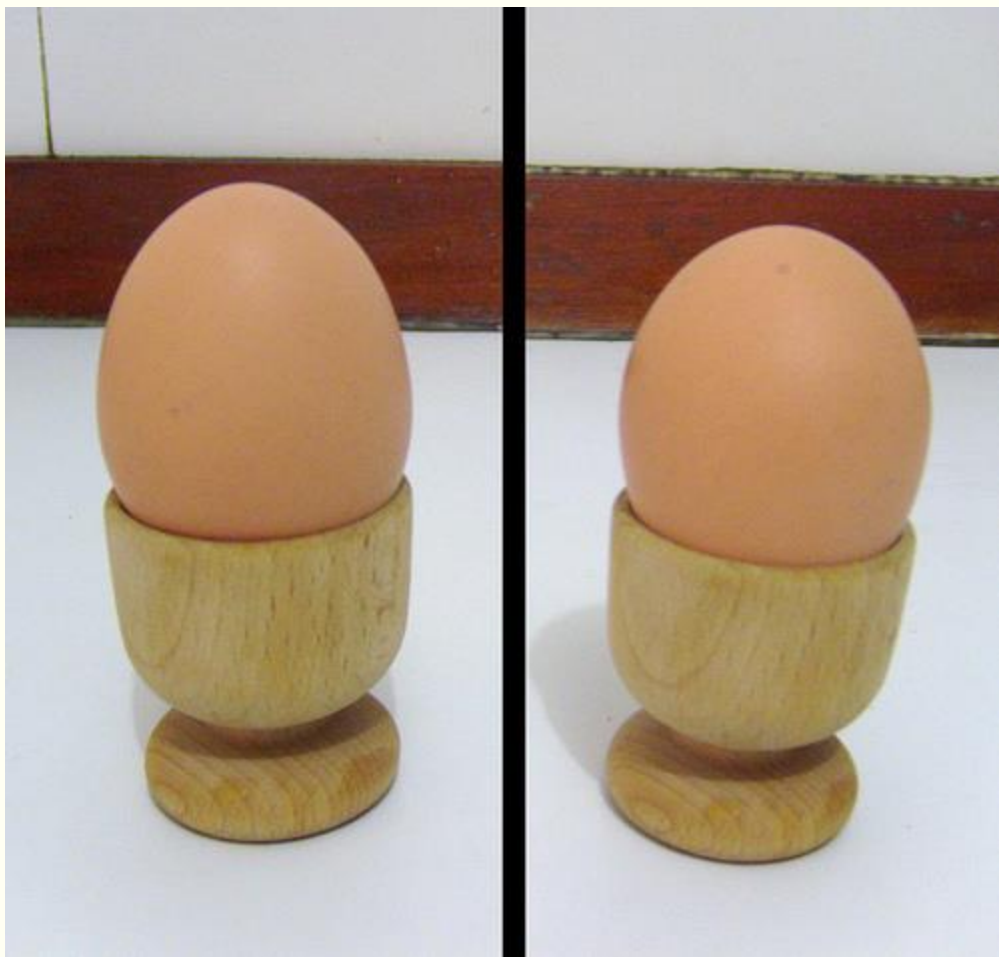


0-й байт	0	1	0	1	1	0	0	0
1-й байт	1	1	0	0	1	1	0	1
2-й байт	1	0	1	0	1	1	1	1
3-й байт	0	0	1	0	1	0	0	1
...								

Межі для представлення цілих чисел

Variety	Bits	Bytes	Minimum	Maximum
unsigned	8	1	0	255
signed	8	1	-128	127
unsigned	16	2	0	65535
signed	16	2	-32768	32767
unsigned	32	4	0	4294967295
signed	32	4	-2147483648	2147483647
unsigned	64	8	0	18446744073709551615
signed	64	8	-9223372036854775808	9223372036854775807

Порядок байтів. Little-Endian проти Big-Endian



- Байт допускає 256 різних значень.
 - Числа представляються одним або декількома байтами A_0, \dots, A_n
 - A_0 - молодший байт числа
 - A_n - старший байт числа
- Порядок від старшого до молодшого (*big-endian* - A_n, \dots, A_0) – звичний порядок запису арабських цифр.
- Порядок від молодшого до старшого (*little-endian* – A_0, \dots, A_n) прийнято для пам'яті в x86-комп'ютерах.
 - Зручніше: при збільшенні розрядності числа розряди дописуються в кінець
 - $3210 \rightarrow 3210'0000$

Навіщо нулю знак?

- Спеціальні числа в IEEE754 (викликають виключення):
 - $\pm\infty$ - отримується від ділення числа на 0, проте при діленні 0/0 – невизначеність (NaN)
 - NaN (not a number) – представляє арифметично беззмістовне значення, $\text{NaN} \neq \text{NaN}$
- Способи отримання NaN:
 - $\infty + (-\infty)$
 - $0 \cdot \infty$
 - $0/0, \infty/\infty$
 - \sqrt{x} , де $x < 0$
- +0 vs -0
 - $+0 = -0$, проте знак збережено для максимальної коректності результату обчислень.
 - Наприклад, $\frac{1}{+\infty} = +0, \frac{1}{-\infty} = -0$
 - $(+\infty/0) + \infty = +\infty$, тоді як $(+\infty/-0) + \infty = \text{NaN}$

Дробові числа у двійковій системі

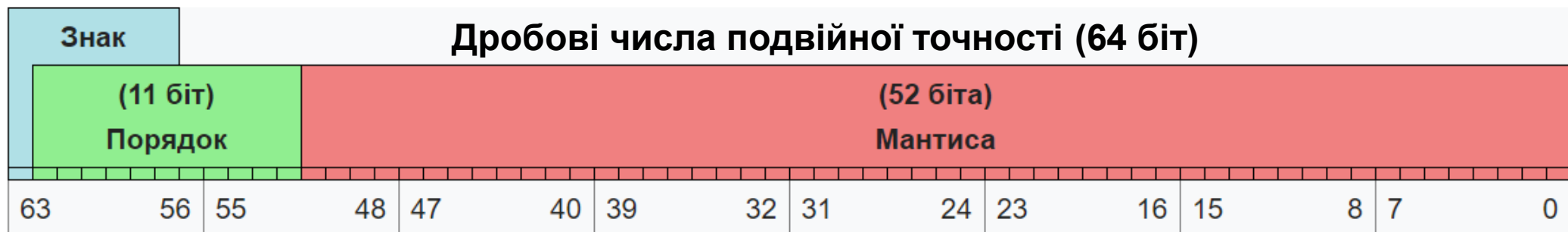


$(-1)^s \cdot M \cdot B^E$,
 s – знак, B – основа,
 E – порядок (експонента),
 M – мантиса

5 = 101 = 1.01e2
 Мантисса = 1.01
 Порядок = 2

**Нормалізоване
представлення**

2.5 = 1*2¹+0*2⁰+1*2⁻¹ = 10.1 = 1.01e1
 Мантисса = 1.01
 Порядок = 1



Дробові числа у двійковій системі

```
Python 3.6 (64-bit)
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 0.1+0.1+0.1==0.3
False
>>> 0.1+0.1+0.1
0.30000000000000004
>>> 0.25+0.25+0.25==0.75
True
```

- Записати дробове десяткове число 0,116 у двійковій системі числення.
 - дробову частину множимо на основу 2, заносючи цілі частини добутку в розряди після коми для шуканого дробового числа:

$$.116 \cdot 2 = \mathbf{0.232}$$

$$.232 \cdot 2 = \mathbf{0.464}$$

$$.464 \cdot 2 = \mathbf{0.928}$$

$$.928 \cdot 2 = \mathbf{1.856}$$

$$.856 \cdot 2 = \mathbf{1.712}$$

$$.712 \cdot 2 = \mathbf{1.424}$$

$$.424 \cdot 2 = \mathbf{0.848}$$

$$.848 \cdot 2 = \mathbf{1.696}$$

$$.696 \cdot 2 = \mathbf{1.392}$$

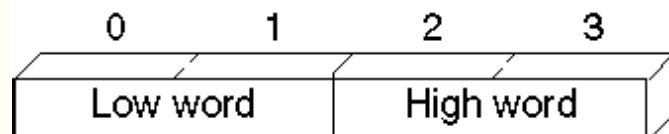
$$.784 \cdot 2 = \mathbf{0.784}$$

і т. д.

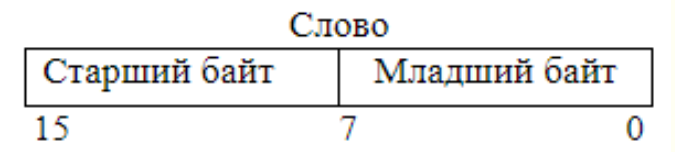
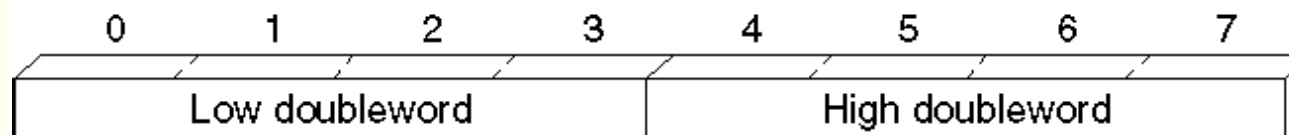
Слова та розрядність

- Для конкретних процесорів одиницями інформації виступають машинні слова.
 - Розрядність даних = довжина стандартного машинного слова.
 - 64-розрядний процесор – 64-розрядне машинне слово
- Іноколи машинне слово приймають як 2 послідовно розташованих байта = 16 біт.
 - 32 біт – подвійне слово (doubleword)
 - 64 біт – збільшене вчетверо слово (quadword)

Doubleword



Quadword



Представлення тексту у двійковій системі. ASCII

Char	ASCII Code	Binary	Char	ASCII Code	Binary
a	097	01100001	A	065	01000001
b	098	01100010	B	066	01000010
c	099	01100011	C	067	01000011
d	100	01100100	D	068	01000100
e	101	01100101	E	069	01000101
f	102	01100110	F	070	01000110
g	103	01100111	G	071	01000111
h	104	01101000	H	072	01001000
i	105	01101001	I	073	01001001
j	106	01101010	J	074	01001010
k	107	01101011	K	075	01001011
l	108	01101100	L	076	01001100
m	109	01101101	M	077	01001101
n	110	01101110	N	078	01001110
o	111	01101111	O	079	01001111
p	112	01110000	P	080	01010000
q	113	01110001	Q	081	01010001
r	114	01110010	R	082	01010010
s	115	01110011	S	083	01010011
t	116	01110100	T	084	01010100
u	117	01110101	U	085	01010101
v	118	01110110	V	086	01010110
w	119	01110111	W	087	01010111
x	120	01111000	X	088	01011000
y	121	01111001	Y	089	01011001
z	122	01111010	Z	090	01011010

- Текст кодується байтовою послідовністю за допомогою таблиці ASCII (American Standard Code for Information Interchange).
 - Перші 128 кодів зарезервували під стандартні символи.
- Утворилось багато кодувань, решта 128 символів у виробників були власними.
 - MS DOS: кодування 855, 866,
 - Windows: 1251,
 - Mac OS використовує своє кодування,
 - KOI8 та KOI7,
 - ISO 8859-5.

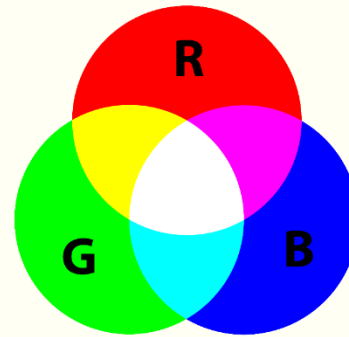
Представлення тексту у двійковій системі. Unicode

- Кожному символу назавжди присвоюється певний код – **кодова точка**.
 - Перші 256 символів відповідають ASCII-таблиці стандарту [ISO 8859-1](#).
 - UTF-16 – кожному символу з [базової багатомовної площини](#) встановлюється 16-бітний номер.
 - Решта символів – по 4 байта (32 біти).
- Проблема – несумісність з ASCII (символи двобайтні, а не одnobайтні).
 - UTF-8 включає одnobайтні символи ASCII (256 штук)
 - Далі – двобайтні символи (більшість європейських алфавітів, іврит, арабський алфавіт)
 - Три байти – базова багатомовна площина
 - Чотири байти – решта символів
- Робота з UTF-8 складна, проте текст
 - компактний,
 - містить захист від помилок зчитування
 - інтернаціональний: для всіх виглядатиме однаково.

character	encoding	bits
A	UTF-8	01000001
A	UTF-16	00000000 01000001
A	UTF-32	00000000 00000000 00000000 01000001
あ	UTF-8	11100011 10000001 10000010
あ	UTF-16	00110000 01000010
あ	UTF-32	00000000 00000000 00110000 01000010

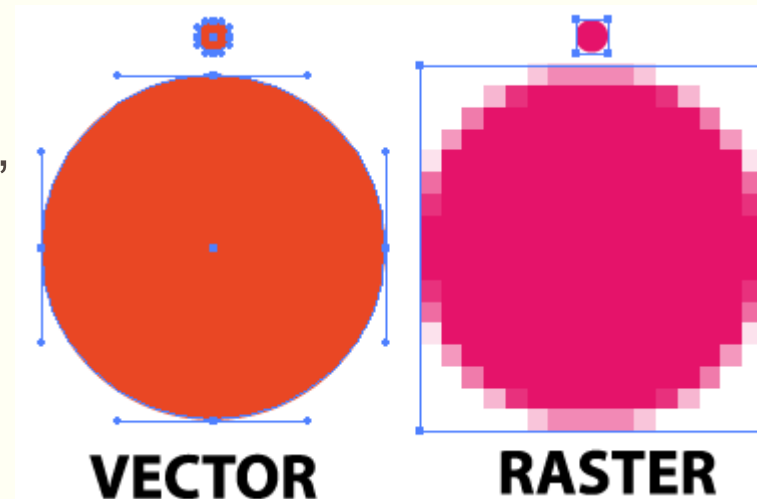
Представлення зображень та відео у двійковій системі

- Кожен піксель кодується кількома цілими числами.

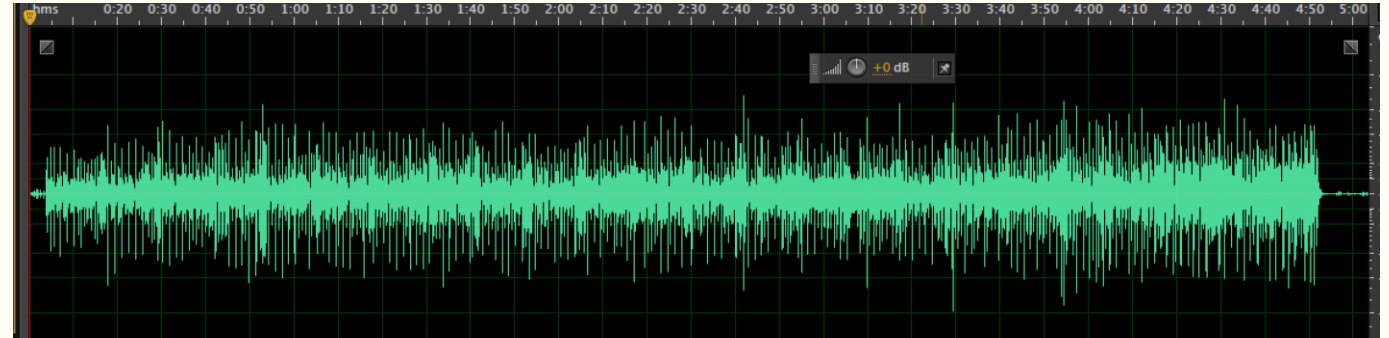
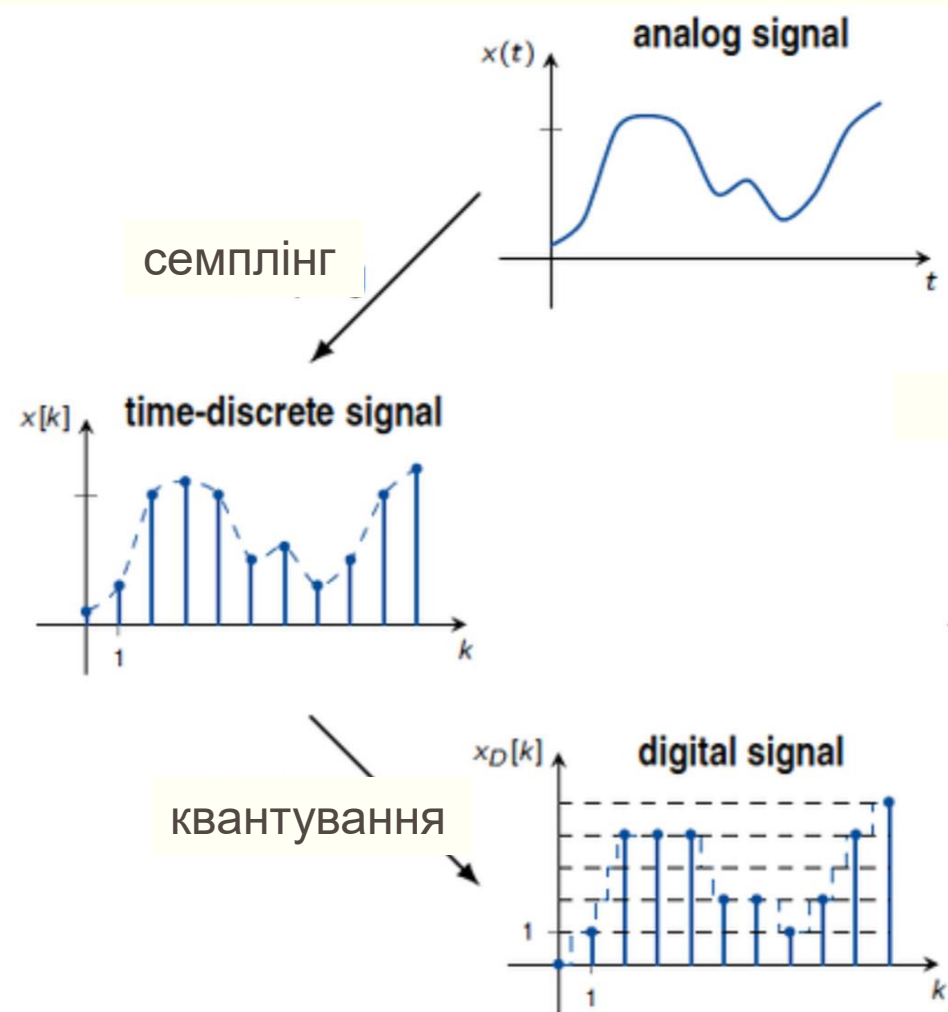


Представлення зображень та відео у двійковій системі

- Для векторної графіки характерне розбиття зображення на ряд графічних примітивів – точки, прямі, ламані, дуги, полігони.
 - Для відрисовки прямої потрібні координати двох точок, які зв'язуються найкоротшою прямою лінією.
 - Для дуги задається радіус і т. д.
 - Зображення зберігається в пам'яті як база даних описів примітивів.
- Відео – набір растрових зображень (кадрів).
 - Растрове зображення – набір чисел.



Представлення аудіо у двійковій системі



```
Audio
Format           : MPEG Audio
Format version    : Version 1
Format profile    : Layer 3
Duration          : 3 min 44 s
Bit rate mode     : Constant
Bit rate          : 320 kb/s
Channel(s)        : 2 channels
Sampling rate     : 44.1 kHz
Frame rate        : 38.281 FPS (1152 SPF)
Compression mode  : Lossy
Stream size       : 8.57 MiB (99%)
```



Тема доповіді: аналогово-цифрове та цифро-аналогове перетворення звуку

Шістнадцяткова система числення

- Використовується програмістами для опису місця знаходження в пам'яті кожного байту.
 - Замість 8 бітів або трьох десяткових розрядів – два шістнадцяткових розряди.
 - Простіша для розуміння, порівнюючи з двійковою системою числення.
 - Зазвичай до шістнадцяткових чисел дописують префікс 0x або суфікс h
- Набагато простіше представляється в пам'яті комп'ютера, порівнюючи з десятковою системою.
 - 4 біти замінюються на одне шістнадцяткове значення.
 - $(1110)_2 = 0\text{Eh}$
 - $(10011100)_2 = 0\text{x9C}$
- Використовується для короткого запису кольорів (R, G, B)
 - #23A1F2

Бінарна	Шістнадцяткова
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Загалом про системи числення

10	2	8	16
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

■ Яке з чисел більше?

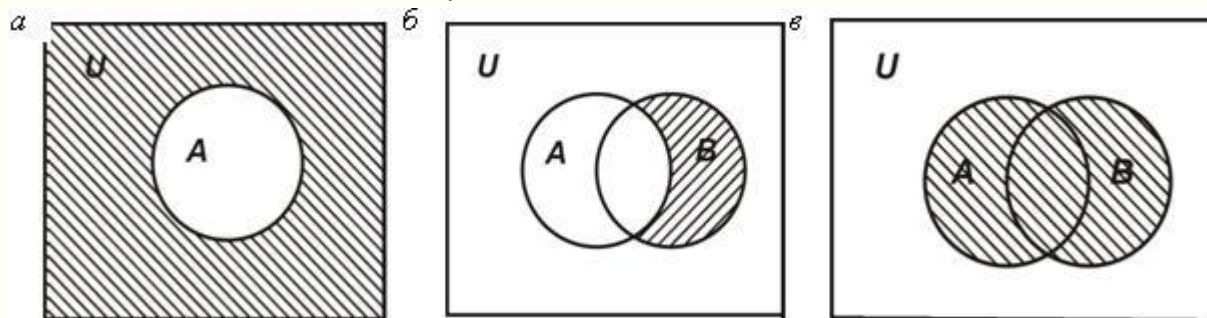
- $9B_{16}$
- 10011010_2
- 153
- 234_8

Base	Readable	Converts easily into bytes	Good compression
4	✓	✓	
8	✓		
16	✓	✓	✓
32	✓		✓
64	✓		✓
128			✓
256		✓	✓

Двійкова логіка та логічні елементи

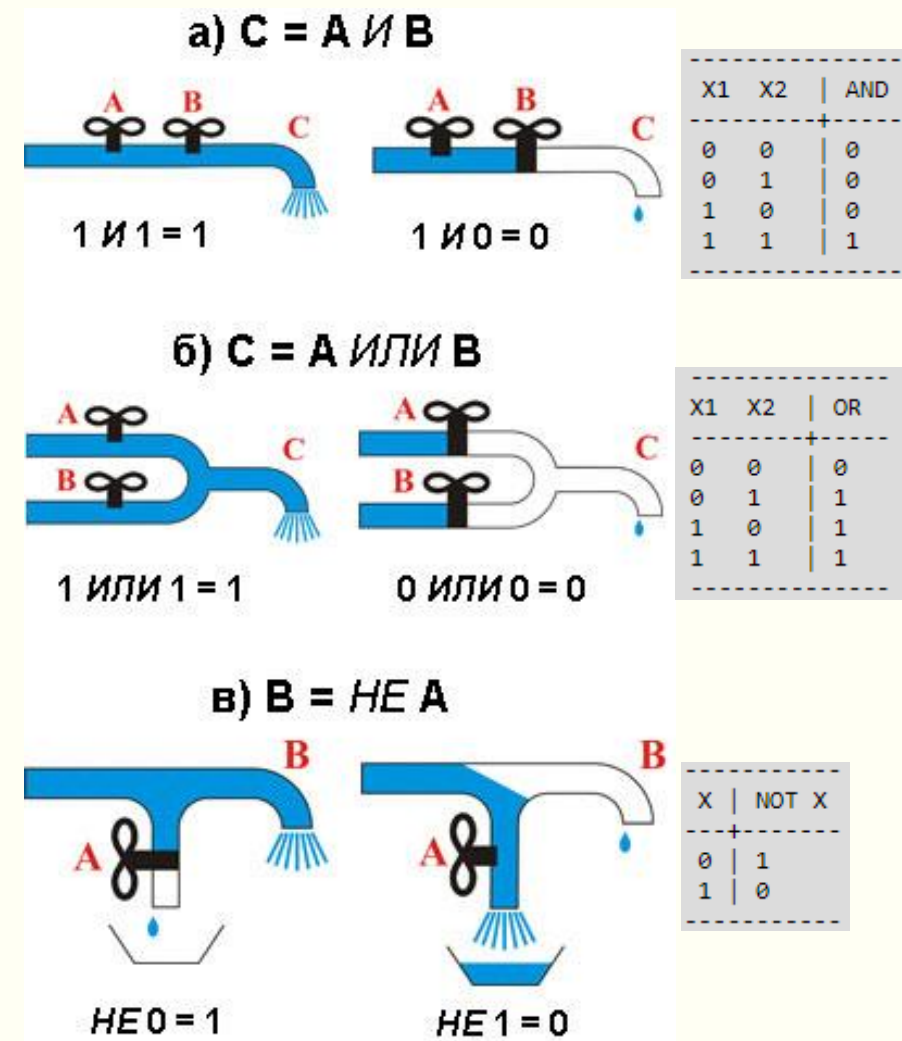
- Основними логічними операціями є:

- Унарна операція заперечення (інверсія, операція НЕ, \bar{A} , $\neg A$)
- Бінарна операція логічного множення (кон'юнкція, операція І, $A \cdot B$, $A \wedge B$)
- Бінарна операція логічного додавання (диз'юнкція, операція АБО, $A \vee B$, $A + B$)



- Операції можуть комбінуватись, наприклад:

- І-НЕ (логічний елемент Шефера)
- АБО-НЕ (логічний елемент Пірса)
- Виключна диз'юнкція (\oplus) – заперечення логічної еквівалентності ($A \leftrightarrow B$)



Логічні елементи та їх позначення

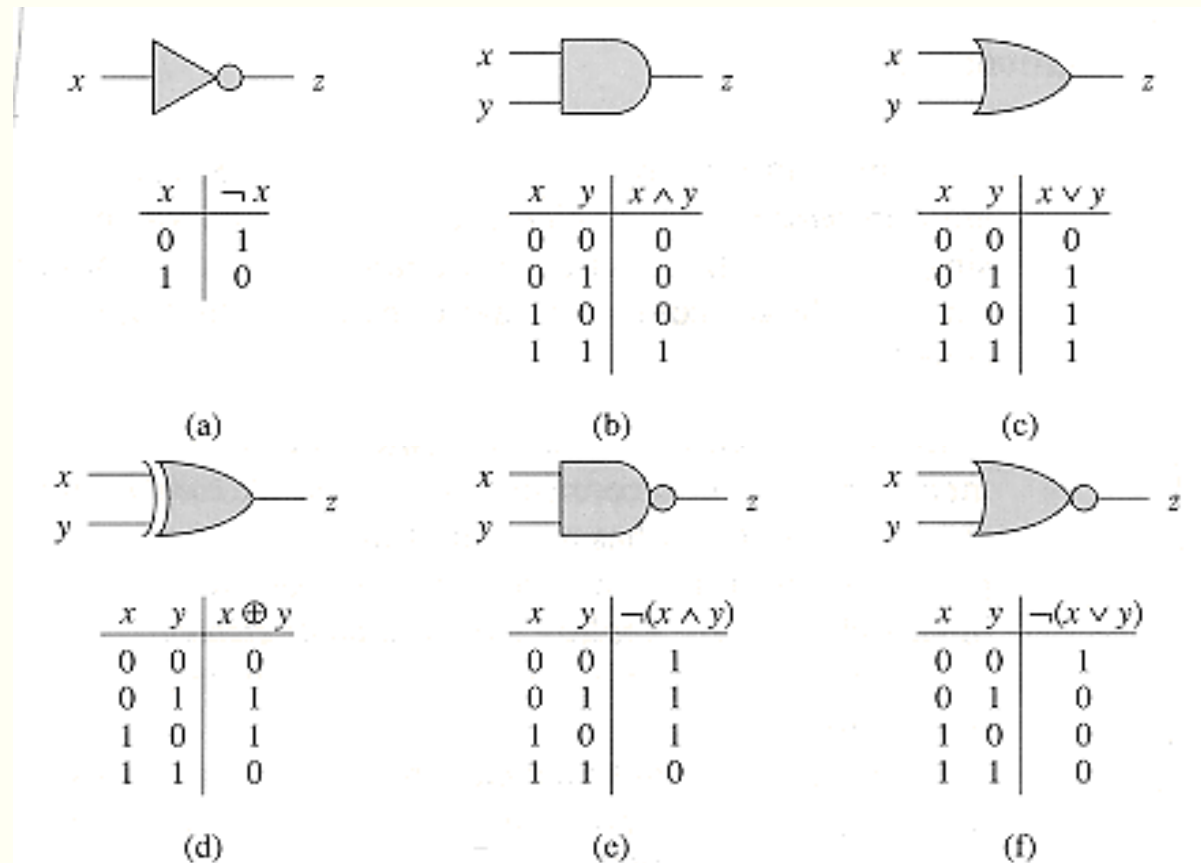
- Існує кілька видів схемотехнічних позначень логічних елементів:

- ANSI (американські позначення, використовуються в лекції)
- DIN (європейські позначення)

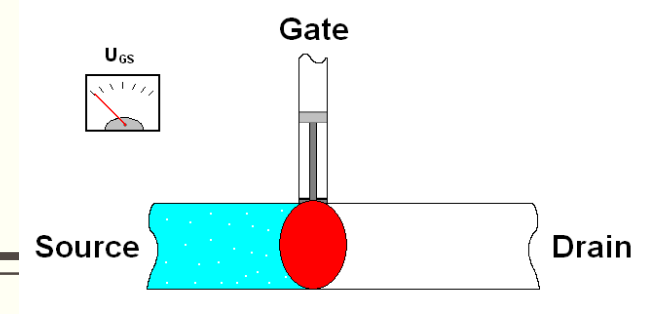
- Логічні елементи:

- a) НЕ = NOT
- b) Логічне І = AND
- c) Логічне АБО = OR
- d) Виключна диз'юнкція = XOR
- e) І-НЕ = NAND
- f) АБО-НЕ = NOR

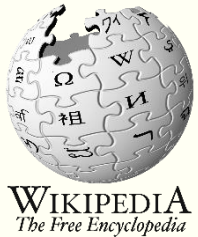
- XNOR???**

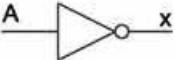








Логічні вентиля (logic gates)



- **Логічний вентиль** — базовий елемент цифрової схеми, що виконує (обчислює) елементарну логічну операцію, перетворюючи таким чином вхідні логічні сигнали у вихідний логічний сигнал.



Name	NOT	AND	NAND	OR	NOR	XOR	XNOR																																																																																																
Alg. Expr.	\overline{A}	AB	\overline{AB}	$A + B$	$\overline{A + B}$	$A \oplus B$	$\overline{A \oplus B}$																																																																																																
Symbol																																																																																																							
Truth Table	<table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	X	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	1
A	X																																																																																																						
0	1																																																																																																						
1	0																																																																																																						
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					

Практична реалізація

Алгебраїчні операції у двійковій системі. Додавання

- Правила додавання:

- $0 + 0 = 0$
- $1 + 0 = 1$
- $0 + 1 = 1$
- $1 + 1 = 10$ (два вентилі, AND і XOR)

- Приклад

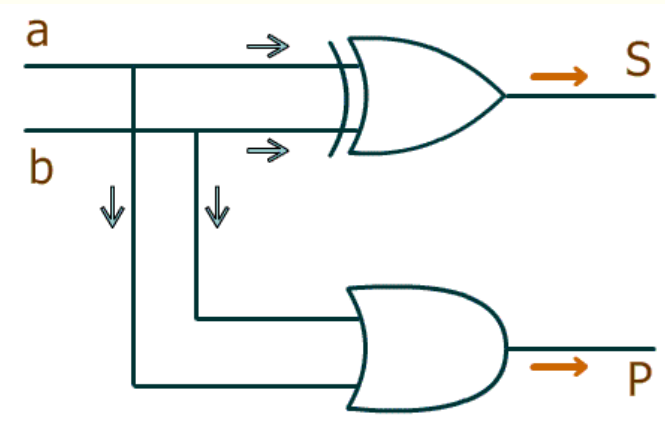
Diagram illustrating the iterative steps of binary addition for 1110 + 0111:

$$\begin{array}{r} 1110 \\ + 0111 \\ \hline 10101 \end{array} \rightarrow \begin{array}{r} 1110 \\ + 0111 \\ \hline 1 \end{array} \rightarrow \begin{array}{r} 1110 \\ + 0111 \\ \hline 01 \end{array}$$

The green arrow indicates the carry (1) from the first addition is used in the second addition.

$$\begin{array}{r} 1110 \\ + 0111 \\ \hline 101 \end{array} \rightarrow \begin{array}{r} 1110 \\ + 0111 \\ \hline 10101 \end{array}$$

INPUTS		OUTPUTS	
A	B	P	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



P – перенесення в наступний розряд
S – сума розряду

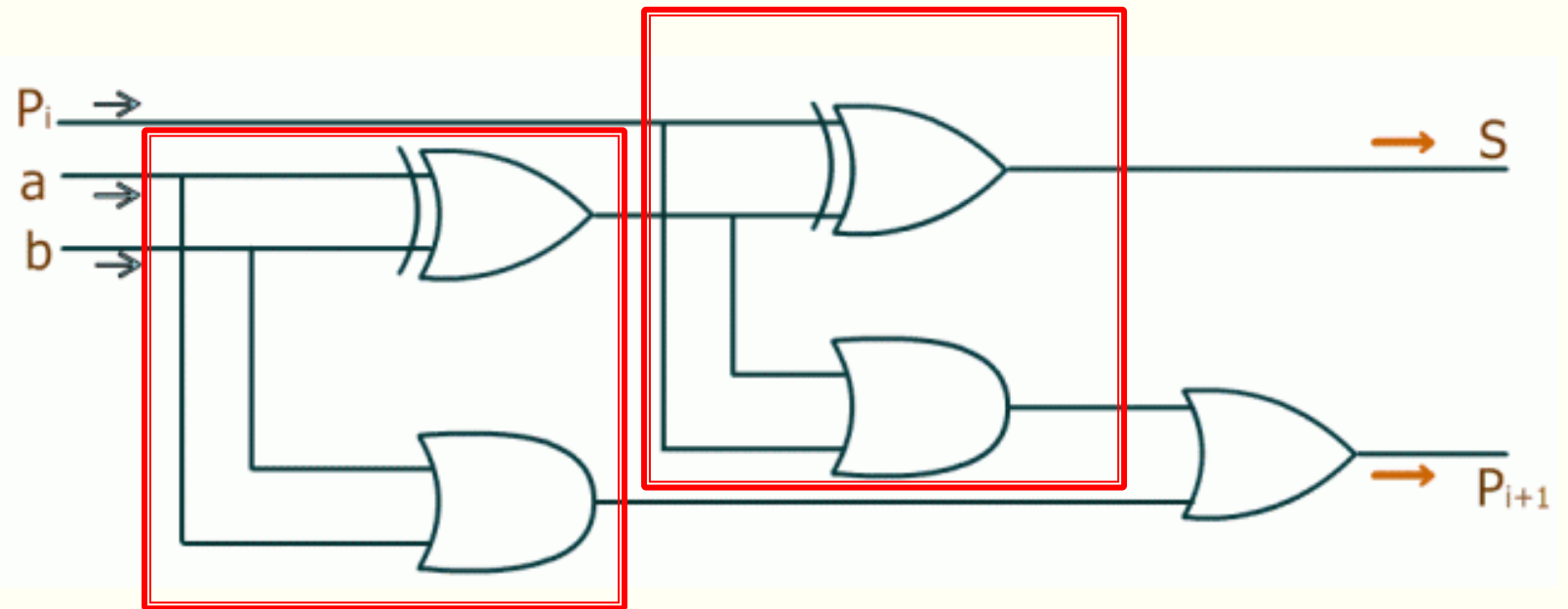
- Схема називається напівсуматором (half adder).

- Значний недолік – не враховує можливе перенесення одиниці з **молодшого** розряду

Додавання у двійковій системі. Суматор (full adder)

- Для врахування переносу з молодшого розряду треба організувати 3 входи та поєднати два напівсуматора:

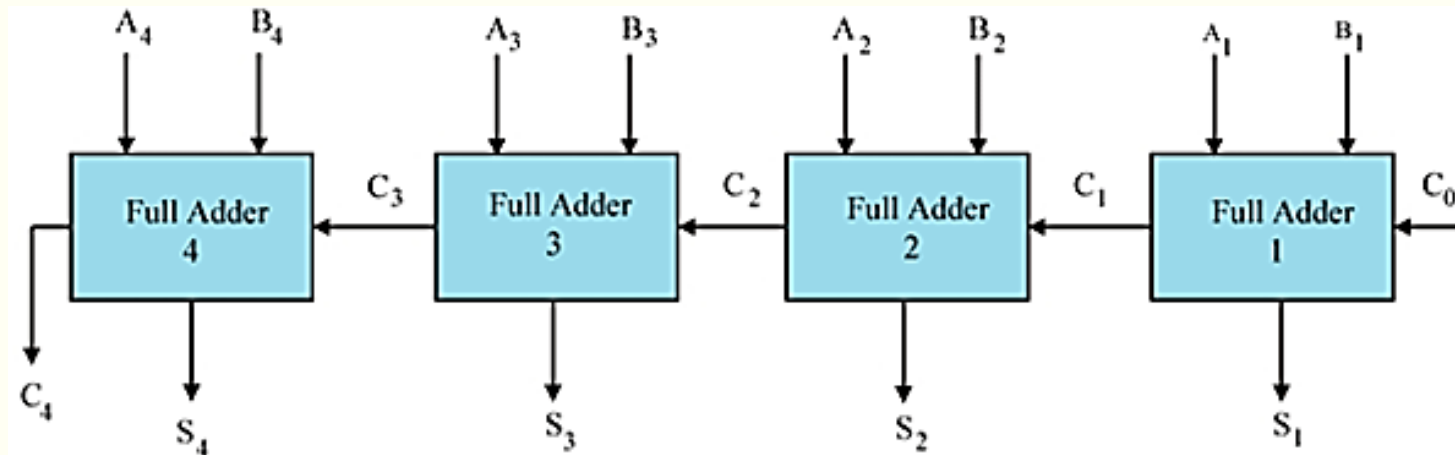
Входи			Выходы	
a	b	P_i	S	P_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
1	0	0	1	0
0	1	1	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



- Такий суматор називають **однобітним** (One-bit Full Adder).

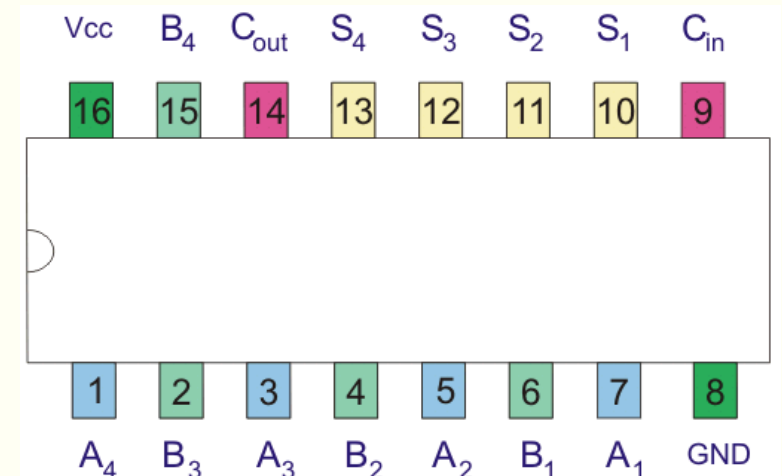
Мультибітове додавання

- Потребує каскад суматорів, з'єднаних в ланцюг за допомогою carry-ліній



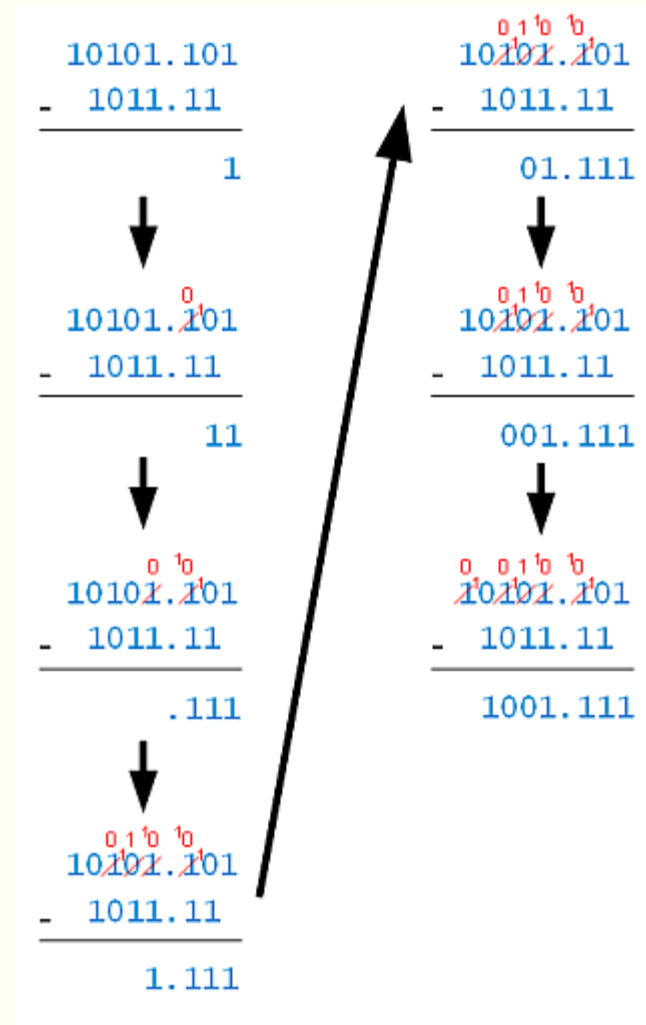
$$\begin{array}{r} 1011 \\ 1101 \\ \hline 11000 \end{array}$$

Here, $A_1 = 1, A_2 = 1, A_3 = 0, A_4 = 1.$
 $B_1 = 1, B_2 = 0, B_3 = 1, B_4 = 1.$



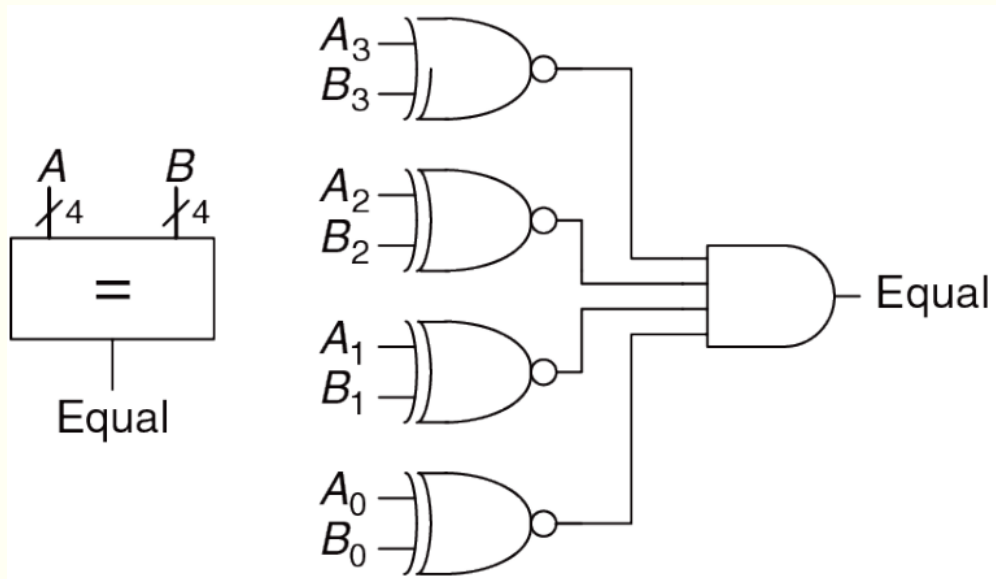
Віднімання у двійковій системі

- Правила віднімання:
 - $0 - 0 = 0$
 - $1 - 0 = 1$
 - $0 - 1 = (\text{займаємо із старшого розряду}) 1$
 - $1 - 1 = 0$
- $x - y = x + (-y)$
 - $-y = (\neg y) + 1$
- $x - y = x + (\neg y) + 1$
 - Суму може обчислити суматор

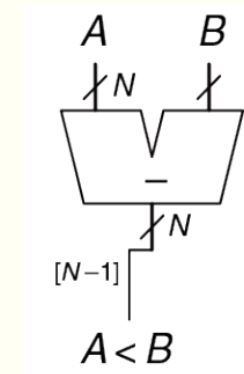


Компаратори

- Компаратори визначають, чи є два двійкових числа рівними або одне з них більше/менше за інше.
 - Компаратор отримує два N-розрядних двійкових числа A і B.
 - Компаратор рівності видає один вихідний сигнал, показуючи, чи рівні A і B ($A=B$).
 - Компаратор величини видає один або більше вихідних сигналів, показуючи відношення величин A і B.



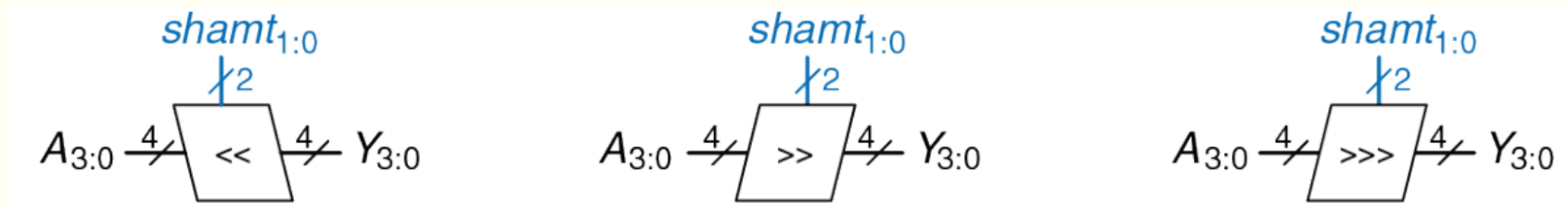
4-розрядний компаратор рівності



N-розрядний компаратор величини

Схеми зсуву і циклічного зсуву

- Переміщують біти (множать або ділять на степінь 2).
 - **Логічні схеми зсуву** зсувають число вліво (LSL) або вправо (LSR) і заповнюють порожні розряди нулями: $11001 \text{ LSR } 2 = 00110$; $11001 \text{ LSL } 2 = 00100$
 - **Арифметичні схеми зсуву** діють так же, проте при зсуві вправо вони заповнюють найбільш значущі розряди значенням знакового біта початкового числа. (необхідно при множенні та діленні знакових чисел): $11001 \text{ ASR } 2 = 11110$; $11001 \text{ ASL } 2 = 00100$
 - **Схеми циклічного зсуву** зсувають число по колу так, що порожні місця заповнюються розрядами, що зсунуті з іншого кінця: $11001 \text{ ROR } 2 = 01110$; $11001 \text{ ROL } 2 = 00111$.



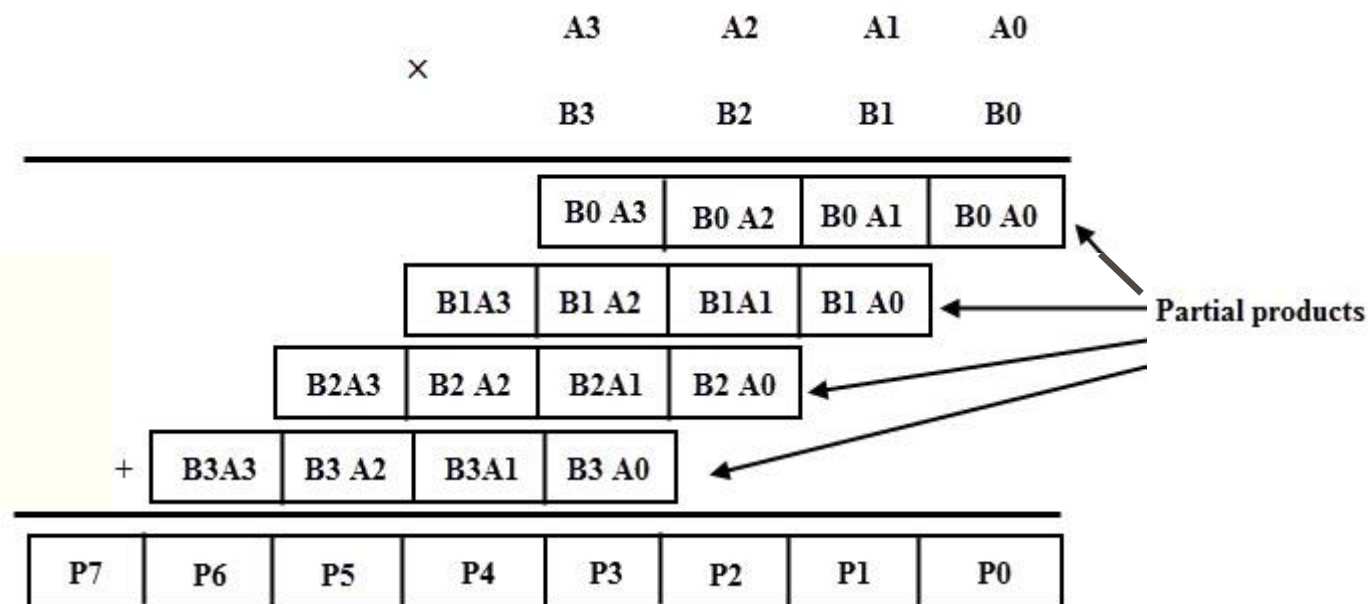
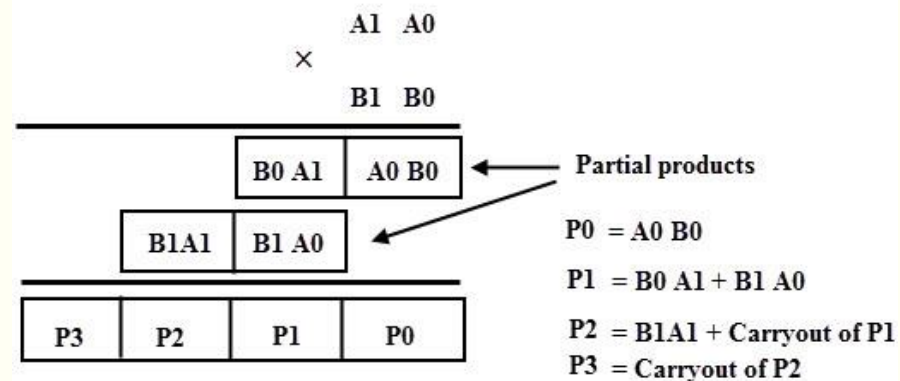
4-розрядні схеми зсуву: (а) зсув уліво, (б) логічний зсув управо, (с) арифметичний зсув вправо

Множення у двійковій системі

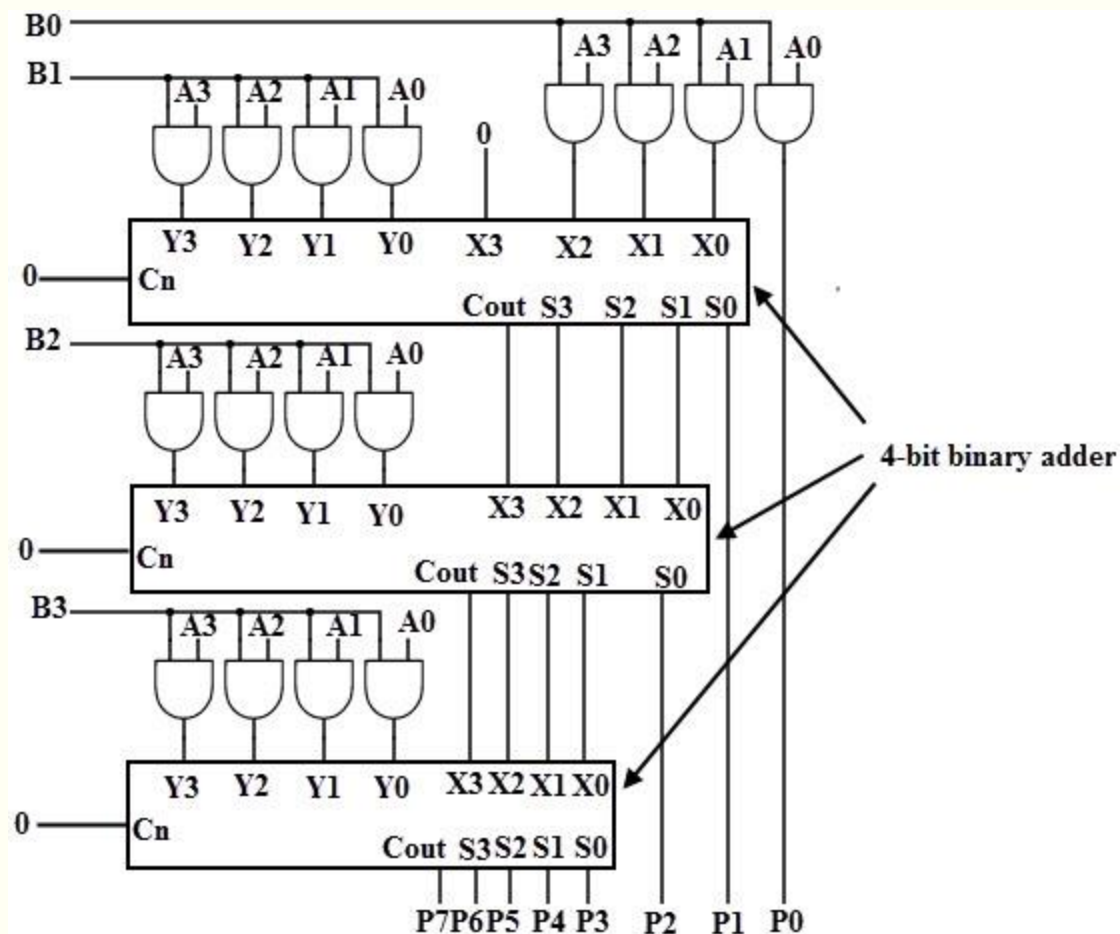
Правила множення:

- $0 * 0 = 0$
- $1 * 0 = 0$
- $0 * 1 = 0$
- $1 * 1 = 1$

1 0 1 0	→	Multiplicand
× 1 0 1 1	→	Multiplier
1 0 1 0	→	Partial product 1
1 0 1 0	→	Partial product 2
0 0 0 0	→	Partial product 3
1 0 1 0	→	Partial product 4
1 1 0 1 1 1 0		



Множення у двійковій системі



- Існують інші алгоритми множення.

Multiplicand:	1011
Multiplier:	× 1010
Initialize Partial Product Z to 0	0000
Add ($Y_0 = 0$) × ($X = 1011$) to Z and shift right one position	<u>+ 0000</u> 00000
Add ($Y_1 = 1$) × ($X = 1011$) to Z and shift right one position	<u>+ 1011</u> 010110
Add ($Y_2 = 0$) × ($X = 1011$) to Z and shift right one position	<u>+ 0000</u> 0010110
Add ($Y_3 = 1$) × ($X = 1011$) to Z and shift right one position	<u>+ 1011</u> Product: 01101110

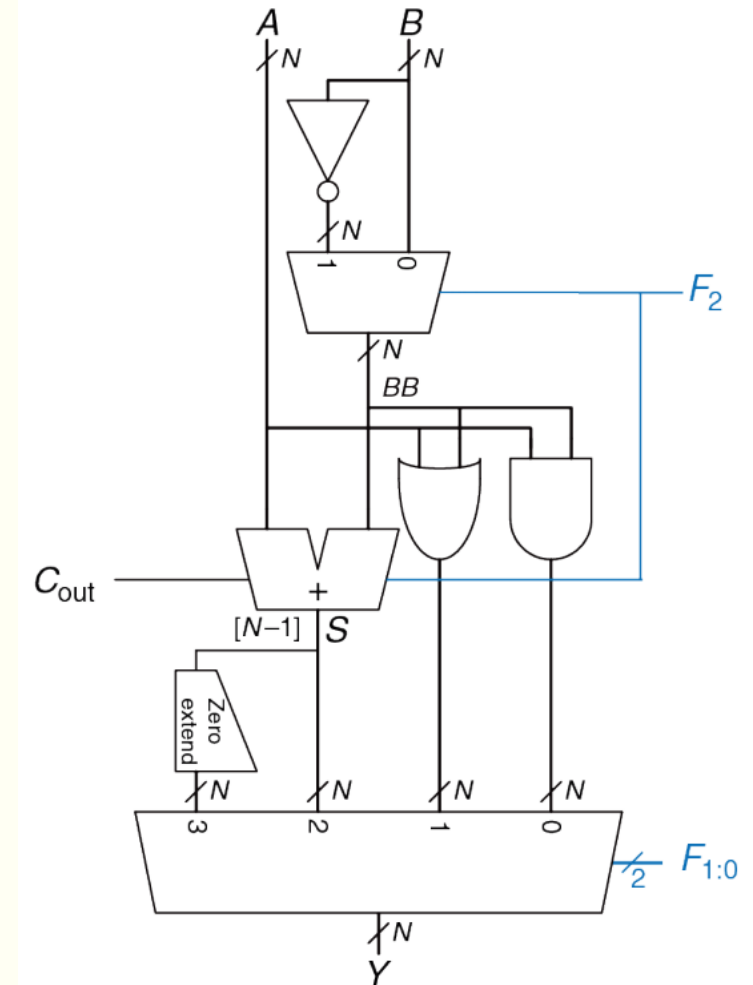
Ділення в двійковій системі

	Divisor D:	00110101
		00000101
	Zero/Partial Remainder R:	0000000000110101
Shift R and Q left. Subtract D. Since the new R is –, replace with prior R, and append 0 to Q.		000000000110101
	–	<u>00000101</u> Quotient ↓
		0000000001101010
Shift R and Q left. Subtract D. Since the new R is –, replace with prior R, and append 0 to Q.		000000001101010
	–	<u>00000101</u>
		0000000011010100
Shift R and Q left. Subtract D. Since the new R is –, replace with prior R, and append 0 to Q.		000000011010100
	–	<u>00000101</u>
		0000000110101000
Shift R and Q left. Subtract D. Since the new R is –, replace with prior R, and append 0 to Q.		000000110101000
	–	<u>00000101</u>
		0000001101010000
Shift R and Q left. Subtract D. Since the new R is +, append 1 to Q.		000001101010000
	–	<u>00000101</u>
		0000000110100001
Shift R and Q left. Subtract D. Since the new R is –, replace with prior R, and append 0 to Q.		000000110100001
	–	<u>00000101</u>
		0000001101000010
Shift R and Q left. Subtract D. Since the new R is +, append 1 to Q.		000001101000010
		0000000110000101
Shift R and Q left. Subtract D. Since the new R is –, replace with prior R, and append 0 to Q.		000000110000101
	–	<u>00000101</u> Quotient ↓
	Remainder:	0000001100001010

Найпростіший АЛУ

$F_{2:0}$	Function
000	A AND B
001	A OR B
010	A + B
011	not used
100	A AND \bar{B}
101	A OR \bar{B}
110	A - B
111	SLT

- Складається з N -бітного суматора, N двовходових логічних елементів І та двоходових логічних елементів АБО.
 - Також містить інвертори та мультиплексор для інверсії бітів B , коли управляючий сигнал F_2 активний.
 - Мультиплексор з організацією 4:1 обирає необхідну функцію на основі сигналів управління $F_{1:0}$.



Quiz

▪ Скільки бітів потрібно для представлення числа 1121?

A. 6

D. 11

B. 8

E. 12

C. 10

F. 16

▪ Що виведеться в консолі після виконання інструкції $0.125+0.125+0.125==0.375$?

A. 0

C. True

B. 0.375

D. False

▪ Колір (15, 0, 240) у двійковому коді з порядком *little-endian* запишеться як

A. 000011110000111100001111

C. 000011111111000011110000

B. 000011110000000011110000

D. 111100000000000011110000

Теми доповідей



Тема доповіді: фізичні принципи роботи сучасних транзисторів



Тема доповіді: проектування логічних схем



Тема доповіді: арифметика з плаваючою крапкою



Тема доповіді: алгоритми множення у комп'ютерних системах



Тема доповіді: суматори



ДЯКУЮ ЗА УВАГУ!

Наступне запитання: мова програмування та трансляція програмного коду