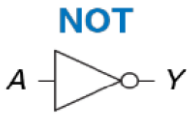
На цій лекції ми розглянемо логічні елементи та їх логіку, а також які логічні операції використовуються у Асемблері

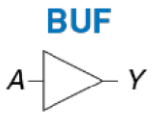
Логічні елементи, або правильніше казати **логічні вентилі** (logic gates) – це найпростіші цифрові схеми, які одержують один або більше двійкових сигналів на вході та виробляють новий двійковий сигнал на виході.
При графічному зображенні логічних вентилів позначення одного чи кількох вхідних сигналів і вихідного сигналу використовуються спеціальні символи. Якщо дивитися зображення логічного елемента, то вхідні сигнали зазвичай розміщуються ліворуч (чи згори), а вихідні сигнали – справа (чи знизу). Розробники цифрових систем зазвичай використовують перші літери латинського алфавіту для позначення вхідних сигналів та латинську букву Y для позначення вихідного сигналу.
Взаємозв'язок між вхідними сигналами та вихідним сигналом логічного вентиля може бути описаний за допомогою **таблиці істинності** (truth table) або рівнянням Булевої логіки. Зліва таблиці істинності представлені значення вхідних сигналів, а праворуч – значення відповідного вихідного сигналу. Кожен рядок у такій таблиці відповідає одній із можливих комбінацій вхідних сигналів.
Рівняння Булевої логіки - це математичне вираз, що описує логічний елемент за допомогою бінарних змінних.



$Y = \bar{A}$

A	Y
0	1
1	0

Логічний вентиль НІ (NOT gate) має один вхід A і вихід Y, як показано на Рис. 1. Причому вихідний сигнал Y - це сигнал, зворотний вхідному сигналу A, або, як кажуть, інвертований A (inversed A). Якщо сигнал на вході A – це БРЕХНЯ, сигнал на виході Y буде ІСТИНА. Таблиця істинності та рівняння Булевої логіки на Рис. 1 підсумовують цей зв'язок вхідного та вихідного сигналів. У рівнянні Булевої логіки лінія над позначенням сигналу читається як «не», тобто математичний вираз $Y = \bar{A}$ вимовляється як «Y дорівнює не A». Саме тому логічний вентиль також називають інвертором (inverter). Для позначення логічного вентиля НІ використовуються також і інші способи запису, включаючи:
$Y = A', Y = \neg A, Y = !A$ і $Y = \sim A$.

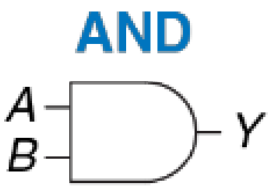


$Y = A$

A	Y
0	0
1	1

Іншим прикладом логічного вентиля з одним входом є **буфер (buffer)**, показаний Рис. 2. Буфер просто копіює вхідний сигнал на вихід. Якщо розглядати буфер як частину логічної схеми, такий елемент нічим не відрізняється від простого дроту і може здатися марним. Разом з тим, на аналоговому рівні буфер може забезпечити характеристики, необхідні для нормального функціонування пристрою, що розробляється. Буфер, наприклад, необхідний передачі великого струму електродвигуну чи швидкої передачі сигналу відразу кільком логічним елементам. Тобто це приклад, який доводить необхідність розгляду будь-якої системи з кількох точок зору, якщо хочемо повною мірою зрозуміти цю систему. Розгляд буфера лише з позиції цифрового рівня не дозволяє розглянути його реальну функцію.
У логічних схемах буфер позначається трикутником. Значок круга на виході логічного елемента, в англомовній літературі, часто званий бульбашкою (bubble), вказує на інверсію сигналу, як, наприклад, показано на рисунку вентиля НІ (Рис. 1).

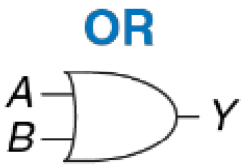
Логічні вентилі з двома вхідними сигналами набагато цікавіші, ніж вентиль НІ та буфер.



$Y = AB$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Логічний вентиль І (AND gate), показаний на Рис. 3 видає значення ІСТИНА на вихід Y виключно тільки якщо обидва вхідні сигнали А і В мають значення ІСТИНА. В іншому випадку вихідний сигнал Y має значення брехня. У таблиці вхідні сигнали перераховані в порядку 00, 01, 10, 11, як у випадку підрахунку в двійковій системі числення. «І» може бути записано декількома способами: $Y = A \bullet B, Y = AB,$ або $Y = A \cap B.$ Символ ∩ читається як «перетин» і більше за інших подобається фахівцям у математичній логіці.

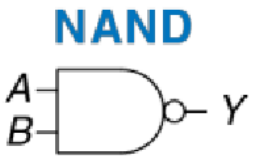


$Y = A + B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Логічний вентиль АБО (OR gate), показаний на Рис. 4 видає значення ІСТИНА на вихід Y, якщо хоча б один з двох вхідних сигналів А або В має значення ІСТИНА. Рівняння Булевої логіки для логічного елемента АБО записується як $Y = A + B$ або $Y = A \cup B.$ Символ ∪ читається як «об'єднання» і знову ж таки найбільше подобається математикам. Розробники цифрових систем зазвичай користуються простим символом +. Математичний вираз Y = A + B звучить "Y дорівнює А або В".

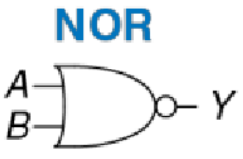
Інші логічні елементи із двома вхідними сигналами



$Y = \overline{AB}$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

На Рис. 5 показані інші поширені логічні вентилі з двома вхідними сигналами. Додавання круга на виході будь-якого логічного вентиля перетворює цей вентиль на нього протилежний – тобто інвертує його. Таким чином, наприклад, з вентиля І виходить вентиль І-НІ (NAND gate). Значення вихідного сигналу Y вентиля І-НІ буде ІСТИНА до тих пір, поки обидва вхідні сигнали А і В не приймуть значення ІСТИНА.



$Y = \overline{A + B}$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Так само з логічного вентиля АБО виходить вентиль АБО-НІ (NOR gate). Його вихідний сигнал Y буде ІСТИНА в тому випадку, якщо жоден із вхідних сигналів, ні А ні В, не має значення ІСТИНА.

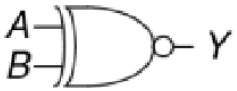
XOR

$Y = A \oplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Виключне АБО з кількістю входів, що дорівнює N (N-input XOR gate) іноді ще називають елементом контролю за парністю (parity gate). Такий вентиль видає на вихід сигнал ІСТИНА, якщо непарна кількість вхідних сигналів має значення ІСТИНА. Як і у випадку елемента з двома вхідними сигналами, комбінації сигналів для елемента з входами N перераховані в логічній таблиці в порядку підрахунку в двійковій системі числення.

XNOR



$Y = \overline{A \oplus B}$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

На Рис. 6 показані позначення і бульове рівняння для вентиля, виключного АБО-НІ (XNOR) з двома входами, який виконує інверсію виключного АБО. Відповідно представлено таблицю істинності. Вихід виключного АБО-НІ є ІСТИНА, якщо обидва входи мають значення БРЕХНЯ або обидва входи мають значення ІСТИНА. Вентиль виключного АБО-НІ з двома входами іноді називають вентилем рівності, так як його вихід є ІСТИНА, коли входи збігаються.

Логічні операції в Assembler

В асемблері x86-32 існує набір інструкцій для виконання операцій над даними, включаючи логічні та булеві операції. Ось деякі з них:

AND (І) інструкція: Виконує логічну операцію І між двома операндами та зберігає результат у першому операнді.

Приклад: AND destination, source

OR (АБО) інструкція: Виконує логічну операцію АБО між двома операндами та зберігає результат у першому операнді.

Приклад: OR destination, source

XOR (Виключне АБО) інструкція: Виконує логічну операцію Виключне АБО між двома операндами і зберігає результат у першому операнді.

Приклад: XOR destination, source

NOT (НІ) Інструкція: Інвертує біти операнда.

Приклад: NOT operand

Важливо не плутати інструкцію NOT з інфструкцією NEG. NEG і NOT - це дві різні інструкції в асемблері x86-32, кожна виконує свою унікальну операцію:

NEG (заперечення): Інструкція NEG виконує операцію заперечення операнда, тобто змінює знак числа на протилежний. Наприклад, якщо у вас є число x, то NEG x перетворить його на -x.

Приклад: NEG destination

NOT (побитове заперечення): Інструкція NOT виконує побитове заперечення для операнда, інвертуючи всі біти в операнді. Якщо біт в операнді був встановлений в 1, він стане 0 і навпаки.

Приклад: NOT destination

Таким чином, основна відмінність між ними полягає в тому, що NEG змінює знак числа, а NOT інвертує всі біти в операнді.

Інструкція TEST: Виконує логічну операцію І між двома операндами, але результат не зберігається. Вона встановлює прапори процесора відповідно до результату операції.

Приклад: TEST operand1, operand2

Коли виконується команда TEST, прапори процесора відображають результат операції (AND) між двома операндами. Різні прапори процесора в архітектурі x86-32 можуть бути встановлені або скинуті в залежності від результату операції.

Zero Flag (ZF): Встановлюється в 1, якщо результат операції AND дорівнює нулю (усі біти результату дорівнюють 0).

Sign Flag (SF): Встановлюється значення найстаршого біта (біта знака) результату. Якщо старший біт дорівнює 1, SF буде встановлено 1, інакше - 0.

Overflow Flag (OF) та Parity Flag (PF): Обидва прапори не визначені для команди TEST і залишаються незмінними.

Carry Flag (CF): Скидається в 0, оскільки операція AND не впливає перенесення між бітами.

Auxiliary Carry Flag (AF): Скидається в 0, оскільки він не визначений для команди TEST.

Direction Flag (DF): Залишається незмінним, оскільки команда TEST не впливає на напрямок операцій введення/виводу.

Interrupt Enable Flag (IF) та Trap Flag (TF): Залишаються незмінними, оскільки команда TEST не впливає на можливість виникнення переривань.

Таким чином, прапори процесора після виконання команди TEST відображатимуть результати операції I (AND) між операндами відповідно до вищеописаних правил.

Приклад:
x = ((-a+b) / (b-c)) / ((c*2+1) * (d/4-1)) *5

```
char a, b
short int c, d
int x

mov
neg
add
sub

sal
sar

inc
dec

imul
idiv
```

CMP (порівняння) інструкція: Виконує віднімання одного операнда з іншого, встановлюючи прапори процесора відповідно до результату операції, але результат не зберігається.
Приклад: CMP operand1, operand2

Ці інструкції можуть бути використані для виконання різних логічних та булевих операцій в асемблері x86-32.