1.3.1 Komputery pracują na danych składających się z zer i jedynek. Można je uporządkować w systemy:

* Binarny (dwójkowy) –podstawowy, oparty na mnożeniu kolejnych potęg liczby 2 przez znak (0/1) na pozycji n-1, gdzie n to potęga liczby 2
* Ósemkowy – zawiera liczby 0-7, nie należy ich mylić z liczbami z systemu dziesiętnego; jeden znak w systemie ósemkowym odpowiada trzem znakom w systemie binarnym.
* Szesnastkowy – zawiera liczby 0-9 oraz litery A-F,  jeden znak w systemie (16) zastępuje  cztery znaki w [systemie binarnym](http://www.korepetycjenowysacz.edu.pl/system-binarny/)

Są to systemy pozycyjne-waga cyfry powiązana z pozycją w ciągu – wagę liczymy **od zera od prawej strony**.

W systemach binarnych również można zapisać liczby ujemne – są dwa sposoby: znak-moduł(ZM) i uzupełnianie do 2. Trzeba oznaczyć takie liczby, żeby było wiadomo, że chodzi o liczbę ujemną. Wtedy też trzeba zarezerwować jeden bit na znak.

*Metoda znak-moduł:* dostawiamy od lewej strony (najstarsza liczba) 0 dla liczby dodatniej i 1 dla ujemnej (z tą liczbą wykonujemy działanie 1-2\*x 🡪 1-2\*1=1-2=-1; 1-2\*0=1-1=0)-jest to tzw znak ; pozostały ciąg zer i jedynek to właściwa liczba binarna.

*Uzupełnienie do 2:* najstarsza cyfra również określa znak liczby, tym razem poprzez mnożenie a\*(-2b), gdzie a to 0 lub 1, b to jej pozycja w danym ciągu. Zamiana z sys. 10 na binarny:

* Wyznaczenie wartości bezwgl danej liczby dziesiętnej
* Zamiana na sys binarny
* Jeśli po zamianie jest nieparzysta ilość zer/jedynek to należy uzupełnić od lewej jedno zero a gdy jest parzysta – dwa zera
* Wykonanie negacji danej liczby →→↘
* Dodać **binarną** jedynkę do powstałej liczby →→ tzw mnemotechniczny algorytm negacji
* (sprawdzenie czy działa)

Liczby stałoprzecinkowe – część całkowita + część ułamkowa; Przy zamianie na sys binarny robimy te części oddzielnie: cz. dziesiętną normalnie, cz. ułamkową w następujący sposób: mnożymy x2-jeśli wyjdzie liczba >1 to wstawiamy 1 i następnie mnożymy x2 tylko część po przecinku a jeśli wyjdzie <1 to wpisujemy o i dalej mnożymy x2; czynność powtarzać aż do momentu, gdy mnożenie zacznie się powtarzać. Jedynki i zera spisujemy od góry do dołu – nie tak jak przy zamianie na sys binarny (gdzie jest od dołu do góry) . Przy zamianie z sys binarnego na dziesiętny wystarczy mnożyć przez potęgi z ujemnymi wykładnikami.

Liczby zmiennoprzecinkowe ?????

Dodawanie liczb binarnych (suma logiczna AND) – podobne do dziesiętnego ale trzeba pamiętać, że:

* 0+0=0
* 0+1=1
* 1+1=2 czyli 1 0 (jedynkę przenosimy „dalej”)

Mnożenie:

* 1\*1=1
* 0\*1=0
* 0\*0=0

Odejmowanie:

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

Inne operacje logiczne: OR (alternatywa), NOT (negacja), XOR (alternatywa wykluczająca), implikacja.

1.3.2 Przerzutniki to układy sekwencyjne – stan wejściowy zależy od stanu wyjściowego (i czasem od stanu wewnętrznego). Zbudowane są w oparciu o bramki logiczne a te z kolei z tranzystorów. Mają kilka wejść (synchroniczne: informacyjne, zegarowe; asynchroniczne: programujące) i zazwyczaj dwa wyjścia.

Tak w skrócie działa przerzutnik: przełącza się pod wpływem sygnałów wejściowych i pamięta ten stan, dopóki inne sygnały wejściowe mu tego stanu nie zmienią.

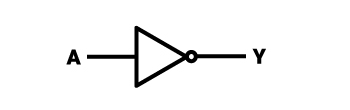
Przerzutniki asynchroniczne przełączają stan wyjść w momencie zmiany na wejściu, a przerzutniki synchroniczne muszą oczekiwać na zmianę sygnału doprowadzonego do ich wejścia zegarowego. W tym momencie należy wyróżnić kolejny podział na przerzutniki wyzwalane zboczem (narastającym lub opadającym) lub wyzwalane poziomem wejścia zegarowego.

Stosowane są do przechowywania małych ilości danych, do których musi być stały dostęp

1.3.3 **Bramki logiczne** to narzędzia stanowiące elementy konstrukcyjne maszyn, automatów czy robotów. Są to elementy elektroniczne, które przyjmują sygnały binarne na wejściach i zwracają wartości 1 lub 0 – prawda lub fałsz. W dzisiejszych czasach najczęściej są to [**układy scalone**](https://botland.com.pl/pl/21-uklady-scalone), składające się z wielu odpowiednio połączonych tranzystorów. W uproszczeniu, zasada działania bramek polega na podaniu jednego z dwóch napięć elektrycznych (poziomów logicznych). Jest to napięcie o wartościach w granicach między 0 V a 0,8 V (poziom 0 – fałsz), natomiast druga wartość napięcia powinna zawierać się w przedziale między 2 V a 5 V (poziom 1 – prawda).

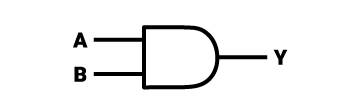
Bramka NOT

Bramka NOT pod względem działania jest najprostsza ze wszystkich tu opisanych (i ogólnie wśród wszystkich wykorzystywanych bramek logicznych). Jej działanie polega na negacji (odwróceniu) sygnału, który otrzyma na wejściu. W praktyce wygląda to następująco: jeżeli na wejściu zostanie nadany sygnał o wartości 1 (prawda), to na wyjściu pojawi się sygnał o wartości 0 (fałsz). Natomiast jeżeli na wejściu pojawi się sygnał 0, na wyjściu pojawi się 1.



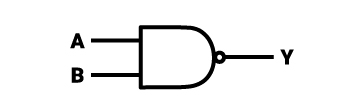
Bramka AND

Przy działaniu bramki logicznej AND wynik 1 można otrzymać tylko w przypadku, kiedy oba wejścia będą równały się jedynce. Tego typu bramki mogą występować w wersjach trzywejściowych, czterowejściowych oraz o znacznie większej liczbie wejść. Należy pamiętać, że niezależnie od tego, ile wejść będzie znajdowało się w stanie wysokim – stan wysoki na wyjściu będzie możliwy tylko w przypadku, jeżeli na każdym wejściu będzie znajdowała się logiczna jedynka.



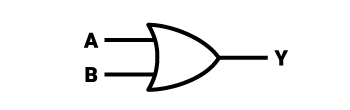
Bramka NAND (-AND)

Dzianie bramki NAND (-AND) jest dokładnie odwrotne do działania bramki AND. Alternatywną nazwą NAND jest określenie “-AND” – myślnik przed oznaczeniem AND oznacza negację. Tak też można interpretować działanie bramki NAND – jest ono dokładnie takie, jakby na wejściu bramki AND zostały umieszczone odpowiednie wartości, na jej wyjściu pojawił się wynik, a następnie został on przepuszczony przez jeszcze bramkę NOT. Można również interpretować działanie NAND w taki sposób, jakby przed każdym z wejść na bramkę AND była bramka NOT. W uproszczeniu działanie bramki NAND można opisać tak – stan niski (0) pojawia się jedynie wtedy, jeżeli na wszystkich wejściach pojawi się stan wysoki (1). Liczba wejść na bramkę NAND jest nieograniczona.



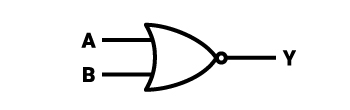
Bramka OR

Na wyjściu bramki OR wynik o wartości 1 pojawi się zawsze w sytuacji, jeżeli chociaż jedno z wejść przyjmuje stan wysoki. Oznacza to również, że jeżeli pojawi się więcej niż jedno wejście o stanie wysokim – na wyjściu również pojawi się jedynka. Tak więc zero pojawi się na wyjściu wyłącznie w sytuacji, kiedy na wszystkich wejściach bramki również ustawione będzie zero. Oczywiście wyjście jest zawsze tylko jedno. Typowy układ zawiera dwa wejścia, ale może posiadać ich nieskończenie wiele i będzie działał na takich samych zasadach.



Bramka NOR

Analogicznie do przedstawionych powyżej bramek, można domyślać się jakiego rodzaju działanie przedstawia bramka NOR. Przeglądając poprzednie nazwy, można zauważyć pewną tendencję: bramki, których nazwy zostaną poprzedzone literą N, będą działały dokładnie odwrotnie w stosunku do swojego oryginału. Litera N (od angielskiego *not*, czyli “nie”) oznacza bowiem negację (jak w przypadku AND i NAND). Działanie bramki NOR można określić jako całkowitą odwrotność działania bramki OR – można spojrzeć na nią, jak na działanie kolejno bramki OR na wszystkie sygnały wejściowe, jakie otrzymała, a następnie wprowadzenie jej sygnału wyjściowego na wejście bramki NOT – wynik będzie wskazywał na działanie bramki NOR. Oczywiście można również założyć, że przed każdym z wejść na bramkę OR zostanie zastosowana bramka NOT – wtedy każde wejście bramki OR będzie pochodziło z wyjścia bramek NOT. Upraszczając opis zasady działania, bramka NOR na wyjściu zawsze zwróci stan niski, z wyjątkiem sytuacji, w której wszystkie stany będą ustawione na wartość 0. Takie bramki mogą mieć nieskończoną ilość wejść.



Bramka XOR (lub inaczej EXOR)

Bramka EXOR (Exclusive-OR, czyli “wyłącznie nie”) to jedna z wyjątkowych funkcji, które nie należą już do grupy najczęściej stosowanych – podstawowych funkcji bramek logicznych. To bramka, która na wejściu ma zawsze dokładnie dwie zmienne (jest to funkcja dwóch zmiennych). Uzyskuje wysoki stan zawsze, jeżeli tylko jeden ze stanów wejściowych jest równy logicznej jedynce. Stan niski na wyjściu pojawi się w sytuacji, jeżeli obie wartości na wejściu będą jednakowe. Bramka XOR wykonuje operacja dodawania bitów modulo-2.

