

Bogusi

moją pierwszą książkę poświęcam

Podziękowania

Pragnę w tym miejscu podziękować wszystkim, którzy przyczynili się do powstania tej książki. W szczególności dziękuję panu mgr inż. Zdzisławowi Nowakowskiemu z CKP w Mielcu za współtworzenie pomysłu napisania tej książki oraz cenne uwagi dotyczące jej treści, panu mgr inż. Zdzisławowi Kolanowi z CKP we Wrocławiu za dostarczenie materiałów na temat nowych wersji pamięci i procesorów, pani mgr inż. Zycie Zacharze oraz mojej żonie Bogusi za wnikliwe przeczytanie maszynopisu i uwagi dotyczące treści i formy tekstu, panom Przemkowi Prałatowi i Staszкови Pokutyckiemu za pomoc techniczną.

Chciałbym także podziękować wszystkim moim słuchaczom z Policealnego Studium Zawodowego za wyrozumiałość i współpracę. To dzięki Wam narodził się pomysł napisania tej książki i powstało wiele pomysłów dotyczących jej treści.

Krzysztof Wojtuszkiewicz

Przedmowa

Pisząc książkę *Urządzenia techniki komputerowej. Jak działa komputer*, postawiłem sobie dwa zadania. Pierwsze z nich wynika z mojego przekonania, że komputer mimo swej fizycznej złożoności jest urządzeniem zbudowanym i działającym w sposób bardzo logiczny. Wynika z tego, że jeżeli problem budowy komputera oraz jego działania rozbijemy na wiele prostszych zagadnień, które następnie połączymy w logiczną i spójną całość, to zrozumienie działania komputera nie powinno sprawić kłopotu nawet użytkownikom o zainteresowaniach zdecydowanie humanistycznych. To, czy z zadania tego udało mi się wywiązać, pozostawiam ocenie Czytelników. Z tego też względu będę bardzo wdzięczny za wszelkie uwagi, szczególnie krytyczne, które proszę przysyłać na jeden z niżej podanych adresów. Przyczynią się one do ulepszenia ewentualnych przyszłych wydań niniejszej książki.

Drugie z zadań wiąże się z przedmiotem „Urządzenia techniki komputerowej” wykładanym w policealnym studium zawodowym o specjalności „technik informatyk”. Książka ta jest pomyślana jako pierwsza część dwutomowego podręcznika do tego przedmiotu. Jest ona zgodna z bieżącym programem i pokrywa cały materiał wykładany w ramach pierwszego semestru. Druga część podręcznika znajduje się obecnie w opracowaniu. Pomysł napisania podręcznika wziął się stąd, że przedmiot ten wykładam już od siedmiu lat (od początku powstania specjalności „technik informatyk”). Jednym z problemów, jakie napotkałem przy jego prowadzeniu, był brak jednej książki pokrywającej większość materiału. Potrzebne wiadomości, nawet jeżeli są dostępne, znajdują się w wielu różnych publikacjach. Opracowanie to jest więc próbą ułatwienia pracy zarówno wykładowcom tego przedmiotu, jak i uczniom.

Niniejsza edycja jest drugim wydaniem tej książki. Pierwsze ukazało się pod tytułem *Jak działa komputer PC* w serii „Systemy mikroprocesorowe” wydawnictwa CKP z Wrocławia. W tym wydaniu wprowadzono niewielkie zmiany uaktualniające treść oraz poprawiono kilka dostrzeżonych błędów.

Drugi tom podręcznika będzie dotyczył działania urządzeń peryferyjnych komputera, takich jak napędy dyskowe, monitory, adaptory graficzne i wiele innych. Dla poszczególnych urządzeń zostanie przedstawiona zasada ich działania, parametry oraz sposób komunikacji z systemem, co w sposób naturalny będzie wiązało się z materiałem z pierwszej części.

Na zakończenie pragnę stwierdzić, że dołożyłem wszelkich starań, aby wiadomości zawarte w książce były jak najbardziej aktualne. Elektronika i informatyka są jednak dziedzinami tak szybko rozwijającymi się, że muszę prosić Czytelników o wybaczenie, jeżeli nie znaleźli w książce wyjaśnienia interesujących ich najnowszych zagadnień.

Krzysztof Wojtuszkiewicz

Ewentualną korespondencję proszę nadsyłać na jeden z poniższych adresów:

- Centrum Kształcenia Praktycznego, ul. Strzegomska 49, Wrocław
- Elektroniczne Zakłady Naukowe, ul. Ostrowskiego 22, 53-238 Wrocław
- e-mail: kawojt@masters.ckp.pl

Wstęp

Jak zasygnalizowałem to w przedmowie, w książce staram się w sposób uporządkowany i logiczny przedstawić sposób funkcjonowania komputera i jego podzespołów. Jej rozdziały można podzielić na trzy części.

Rozdziały 1 i 2 zawierają informacje podstawowe potrzebne do zrozumienia funkcjonowania układów cyfrowych, zarówno prostych, jak i bardziej skomplikowanych, pojawiających się w dalszych częściach książki. Wiadomości te, przykładowo systemy liczbowe, kodowanie, arytmetyka dwójkowa czy też działania logiczne, przydatne są także w innych przedmiotach związanych z techniką komputerową, na przykład w językach programowania, ponieważ ułatwiają zrozumienie pewnych operacji czy pojęć.

Rozdział 4 zawiera podstawowe wiadomości o zasadach działania procesora, architekturze komputera i współdziałaniu jego poszczególnych elementów. Zrozumienie i dobre opanowanie materiału tego rozdziału umożliwia spojrzenie w sposób logiczny i uporządkowany na funkcjonowanie komputera, będącego systemem mikroprocesorowym. Część podstawowych wiadomości na temat architektury komputerów, takich jak pamięć podręczna (cache) czy pamięć wirtualna, została wprowadzona w rozdziale 5, gdyż w sposób naturalny wiążą się one z zagadnieniami dotyczącymi funkcjonowania i budowy bardziej skomplikowanych procesorów. Także tu pewne wiadomości, takie jak tryby adresowania czy sposób prezentowania instrukcji mikroprocesora, mogą być przydatne lub ułatwiać zrozumienie pewnych pojęć w innych przedmiotach.

Rozdziały 3, 5, 6 oraz 7 dotyczą struktury płyty głównej oraz podstawowych układów wchodzących w jej skład. W rozdziale 3. zostało przedstawione funkcjonowanie pamięci półprzewodnikowych. Rozdział o pamięciach poprzedza rozdział na temat podstaw architektury komputera, gdyż pewne informacje dotyczące działania i własności pamięci są niezbędne przy omawianiu schematu blokowego systemu mikroprocesorowego (jakim jest komputer). Rozdział 5 przedstawia dość szczegółowo budowę i własności procesorów rodziny Intel 80x86, ze szczególnym uwzględnieniem tych własności, które mają duży wpływ na jakość i sposób działania komputera. Rozdziały 6 i 7 zawierają wiadomości o budowie i funkcjonowaniu płyt głównych oraz zawartych na nich podzespołach, a także o związanych z nimi standardach. Opisano między innymi standard płyty głównej ISA, standardy magistral rozszerzających ze szczególnym uwzględnieniem PCI, a także podstawy standardu Plug and Play umożliwiającego autokonfigurację.

Załączona na końcu książki bibliografia umożliwia odnalezienie bardziej szczegółowych informacji na poszczególne tematy. Odnośniki do poszczególnych pozycji bibliografii znajdują się w tekście książki.

1. Podstawy działania układów cyfrowych

⇒ Ten rozdział

W rozdziale pierwszym przedstawiamy informacje i wprowadzamy pojęcia potrzebne do zrozumienia działania układów cyfrowych, sposobów reprezentowania i przetwarzania w nich informacji. Omawiane są kolejno systemy liczbowe, przykłady kodowania informacji, działania logiczne, podział układów cyfrowych oraz najprostsze układy cyfrowe – bramki i przerzutniki.

⇓ Następny rozdział

W rozdziale drugim omawiamy bardziej rozbudowane układy cyfrowe, zwane układami funkcjonalnymi, ze szczególnym uwzględnieniem tych układów, które mają zastosowanie w technice komputerowej.

Wstęp

Działanie układów cyfrowych oparte jest na wykorzystaniu dwóch stanów elektrycznych tych układów, zwanych stanem niskim (ang. *low* – L) i stanem wysokim (ang. *high* – H). Przy ich użyciu musimy przedstawić wszystkie rodzaje informacji występujące w układach cyfrowych. W tym celu stosowane są określone struktury, takie jak dwójkowy (binarny) system liczbowy oraz kody, umożliwiające reprezentację informacji w układach cyfrowych.

1.1. Systemy liczbowe

1.1.1. System dwójkowy

Ludzie w sposób naturalny przyzwyczajeni są do liczenia w systemie dziesiętnym, dlatego też konstrukcję i użycie systemu dwójkowego przedstawiamy przez analogię do systemu dziesiętnego.

Do zapisu dowolnej liczby bez znaku system dziesiętny wykorzystuje dziesięć symboli graficznych, zwanymi cyframi: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Przy ich użyciu jesteśmy w stanie przedstawić dowolną liczbę. System dziesiętny, podobnie jak i system dwójkowy jest systemem pozycyjnym. Liczbę 425_D (D oznacza zapis liczby w systemie dziesiętnym) możemy przedstawić jako następującą sumę:

$$425_D = 4 * 100 + 2 * 10 + 5 * 1$$

czyli:

$$\begin{array}{ccc} 4 & 2 & 5 \\ \uparrow & \uparrow & \uparrow \\ & & 0 - \text{pozycja jedynek} \\ & & 1 - \text{pozycja dziesiątek} \\ & & 2 - \text{pozycja setek} \end{array} \quad 425_D = 4 * 10^2 + 2 * 10^1 + 5 * 10^0$$

Widzimy więc, że cyfra na danej pozycji mnożona jest przez odpowiednią potęgę liczby 10, przy czym wykładnik tej potęgi zależy od położenia (pozycji) danej cyfry w liczbie. Uwaga! Pozycje cyfr w liczbie numerujemy od 0 (najmłodsza cyfra). Poszczególne mnożniki, zwane inaczej wagami, w systemie dziesiętnym noszą nazwę odpowiednio: jedynek ($10^0 = 1$), dziesiątek ($10^1 = 10$), setek ($10^2 = 100$) i tak dalej. Poszczególne wagi w systemie dziesiętnym są potęgami liczby 10, dlatego jest ona zwana *podstawą* tego systemu ($p = 10$). Podsumowując, formalny zapis $a_{n-1}.....a_0$ w systemie dziesiętnym oznacza:

$$a_{n-1}.....a_{0D} = a_{n-1} * 10^{n-1} + a_{n-2} * 10^{n-2} + + a_0 * 10^0 = \sum_{i=0}^{n-1} a_i * 10^i$$

gdzie i jest numerem pozycji w liczbie, natomiast a_i oznacza dowolną z cyfr od 0 do 9, a n jest ilością cyfr (pozycji) w liczbie.

Ponieważ w systemie dziesiętnym dysponowaliśmy dziesięcioma cyframi dla zapisania dowolnej liczby bez znaku, w systemie dwójkowym musimy do tego celu używać jedynie dwóch cyfr: 0 i 1. Jak już wspomnieliśmy, obydwa systemy są systemami *pozycyjnymi*, co oznacza, że cyfrę na danej pozycji mnoży się przez określoną *wagę*. Dla systemu dwójkowego podstawą jest liczba 2 ($p = 2$) i wagami są odpowiednie potęgi tej liczby. Kolejne pozycje liczby zwane są więc pozycjami jedynek, dwójek, czwórek, ósemek i tak dalej. Zapis w systemie dwójkowym, zwanym inaczej *systemem binarnym*, liczby 10100_B (litera B sygnalizuje liczbę w systemie dwójkowym) oznacza:

$$\begin{aligned} 10100_B &= 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 0 * 2^0 = \\ &= 1 * 16 + 0 * 8 + 1 * 4 + 0 * 2 + 0 * 1 = 16 + 4 = 20_D \end{aligned}$$

Uogólniając, zapis $a_{n-1}.....a_{0B}$ w systemie dwójkowym będzie oznaczał:

$$a_{n-1} \dots a_0_B = a_{n-1} * 2^{n-1} + a_{n-2} * 2^{n-2} + \dots + a_0 * 2^0 = \sum_{i=0}^{n-1} a_i * 2^i$$

Wzór ten, określający sposób zapisu liczby w systemie dwójkowym, pozwala jednocześnie na dokonanie *konwersji* (przeliczenia) liczby zapisanej w systemie dwójkowym na liczbę zapisaną w systemie dziesiętnym.

Jedną z metod konwersji liczby dziesiętnej na dwójkową pokażemy na przykładzie, pomijając uzasadnienie jej poprawności. Metoda ta polega na wykonywaniu kolejnych dzielen całkowitych, z zapisem reszty, przez liczbę 2. Rozpoczynamy od podzielenia liczby przeliczanej przez 2. Kolejne dzielenie wykonujemy na liczbie będącej ilorazem poprzedniego dzielenia. Postępowanie kontynuujemy aż do momentu otrzymania jako wyniku 0. Reszty dzielen ustawione w odpowiedniej kolejności dają poszukiwaną liczbę binarną.

Przykład

Dokonać konwersji liczby 23_D na liczbę binarną.

Rozwiązanie

$23 : 2 = 11 \quad r = 1$	$\uparrow$ kierunek odczytu wyniku
$11 : 2 = 5 \quad r = 1$	
$5 : 2 = 2 \quad r = 1$	
$2 : 2 = 1 \quad r = 0$	
$1 : 2 = 0 \quad r = 1$	

A zatem $23_D = 10111_B$.

1.1.2. System heksadecymalny

System heksadecymalny, czyli szesnastkowy, nie jest używany bezpośrednio przez układy cyfrowe, jest natomiast wygodnym, zwartym sposobem zapisu liczb binarnych. Stosowany jest on często przez programistów czy też przy wyświetlaniu informacji cyfrowej na ekranie (na przykład w programach typu *debugger*).

W systemie heksadecymalnym do zapisu dowolnej liczby dysponujemy szesnastoma cyframi. Ponieważ symboli graficznych oznaczających liczby arabskie jest dziesięć, brakuje symboli sześciu cyfr. Przyjęto więc, że będą one oznaczane początkowymi literami alfabetu (dużymi lub małymi). Zatem A oznacza dziesiątkę, B jedenastkę, aż do cyfry F, która oznacza piętnastkę. Pełny zestaw cyfr heksadecymalnych jest więc następujący:

$$a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$$

gdzie a_i oznacza cyfrę heksadecymalną. Jak łatwo sprawdzić, cyfr heksadecymalnych jest szesnaście. Liczba szesnaście jest też podstawą tego systemu. Formalny zapis $a_{n-1} \dots a_0$ oznacza więc:

$$a_{n-1} \dots a_0 = a_{n-1} * 16^{n-1} + a_{n-2} * 16^{n-2} + \dots + a_0 * 16^0 = \sum_{i=0}^{n-1} a_i * 16^i$$

gdzie a_i jest dowolną cyfrą heksadecymalną.

Przykład

Znaleźć liczbę dziesiętną odpowiadającą liczbie heksadecymalnej $4C2_H$.

Rozwiązanie

Zgodnie z podanym wzorem oraz wcześniejszymi informacjami:

$$\begin{aligned} 4C2_H &= 4 * 16^2 + C * 16^1 + 2 * 16^0 = \\ &= 4 * 256 + 12 * 16 + 2 * 1 = 1218_D \end{aligned}$$

Konwersji liczby dziesiętnej na heksadecymalną można dokonać metodą analogiczną do pokazanej dla systemu dwójkowego, wykonując kolejne dzielenia z resztą przez liczbę 16. Należy jednak pamiętać, że reszty z dzielenia zapisujemy w postaci cyfr heksadecymalnych, czyli np. resztę 14 zapisujemy jako E.

Najistotniejszą cechą systemu heksadecymalnego jest łatwość przechodzenia od zapisu binarnego do heksadecymalnego i na odwrót, przez co zapis heksadecymalny staje się zwartym zapisem liczb binarnych. Pokażemy to na przykładzie.

Przykład

Zapisać liczbę binarną 1001011010_B w postaci liczby heksadecymalnej.

Rozwiązanie

Przy przejściu od liczby binarnej do heksadecymalnej wykorzystujemy fakt, że każdej cyfrze heksadecymalnej odpowiada określona kombinacja czterech cyfr binarnych i na odwrót. Pokazuje to tabela 1.1.

Przeliczaną liczbę binarną dzielimy od końca (czyli od najmłodszej pozycji) na czwórki, a następnie każdą z nich zapisujemy w postaci jednej cyfry heksadecymalnej, zgodnie z tabelą 1.1. Jeżeli ostatni fragment liczby nie jest pełną czwórką, możemy ją dopełnić do czwórki zerami. Tak więc dla liczby binarnej 001001011010 :

$$0010 \mid 0101 \mid 1010_B = 25A_H$$

Podobnie możemy postąpić przy przeliczaniu w drugą stronę. Wówczas każdą cyfrę heksadecymalną zapisujemy w postaci czwórki cyfr binarnych. Ewentualne nieznaczące zera na początku liczby binarnej można w wyniku pominąć.

Tabela 1.1. Cyfry heksadecymalne i odpowiadające im liczby binarne

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

Przykład

Zapisać liczbę heksadecymalną $7cd5_H$ w postaci liczby binarnej.

Rozwiązanie

$$7CD5_H = 0111 \mid 1100 \mid 1101 \mid 0101_B = 111110011010101_B$$

Prosimy porównać długości liczb heksadecymalnych i odpowiadających im liczb dwójkowych. Wyjaśni to wygodę stosowania zapisu heksadecymalnego.

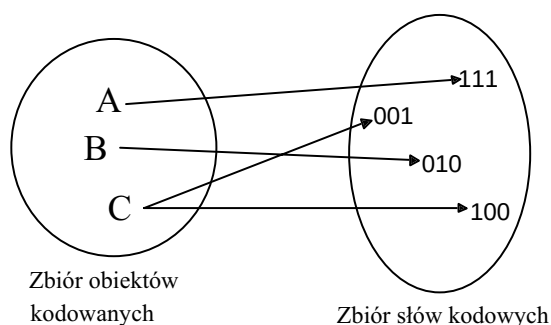
1.2. Kodowanie

Komputer jest urządzeniem służącym do przetwarzania informacji. Informacją są liczby, a także inne obiekty, takie jak litery, wartości logiczne i tym podobne. Ponieważ komputer jest urządzeniem zbudowanym z układów cyfrowych, każda informacja przetwarzana przez niego musi być reprezentowana przy pomocy dwóch stanów – wysokiego i niskiego. Duża część tej informacji to liczby, stąd przyjęło się nazywać te stany *jedynką* i *zerem* (**1** i **0**). Możemy zatem stwierdzić, że wszelka informacja w komputerze musi występować w postaci zerojedynekowej czyli binarnej. Potrzebne są też reguły przekształcania różnych postaci informacji na informację binarną (dokładniejsza definicja informacji binarnej zostanie podana nieco później). Proces przekształcania informacji jednego rodzaju postaci na inną postać nazywamy *kodowaniem*.

Definicja

Kodowaniem nazywamy przyporządkowanie poszczególnym obiektom zbioru kodowanego odpowiadających im elementów zwanych słowami kodowymi, przy czym każdemu słowu kodowemu musi odpowiadać dokładnie jeden element kodowany.

Zbiorem kodowanym może być zbiór dowolnych obiektów, przykładowo liter, symboli graficznych czy np. stanów logicznych. Proces kodowania poglądowo przedstawiony jest na rysunku 1.1.



Rysunek 1.1. Graficzna interpretacja procesu kodowania

Zgodnie z rysunkiem litera A będzie reprezentowana przez słowo kodowe (w skrócie kod) 111, litera B przez 010 zaś litera C przez 001 lub 100. Fakt, że literze C odpowiadają dwa słowa kodowe, nie przeszkadza w poprawnym przetwarzaniu informacji, aczkolwiek stanowi pewne utrudnienie procesu kodowania. Sytuacja odwrotna, gdy jedno słowo kodowe odpowiadałoby dwóm literom (na przykład A – 001 i B – 001), byłaby niedopuszczalna. Jeżeli w procesie przetwarzania informacji otrzymalibyśmy kod 001, nie byłibyśmy w stanie określić przy dekodowaniu, czy w wyniku odpowiada on literze A, czy B.

Sposób określenia kodu, czyli procesu kodowania, może być różnoraki. Może to być opis słowny, wzór, tabela przekodowująca lub każdy inny sposób zapewniający spełnienie warunków podanych w definicji.

Jak wspomnieliśmy, informacja kodowana w komputerze jest bardzo różnorodna. Mogą to być teksty (czyli ciągi znaków), polecenia do wykonania przez komputer (na przykład instrukcje dla procesora), wartości logiczne czy też liczby. W ostatnim przypadku będziemy mówić o tak zwanych *kodach liczbowych*. Będą to kody przedstawiające liczby, z reguły dziesiętne, w postaci binarnej. Poniżej podajemy definicję kodu liczbowego oraz kilka przykładów najczęściej używanych kodów.

Definicja

Kodem liczbowym nazywamy taki kod, który liczbom dowolnego systemu będzie przyporządkowywał słowa kodowe w postaci zerojedynekowej.

Przykład**Naturalny kod binarny (NKB)****Definicja**

Jeżeli dowolnej liczbie dziesiętnej przyporządkujemy odpowiadającą jej liczbę binarną, to otrzymamy naturalny kod binarny (NKB).

Kilka przykładowych wartości liczb kodowanych i odpowiadających im słów kodowych (przy założeniu długości słów kodowych równej 4) zawiera tabela 1.2.

Tabela 1.2. Przykłady słów kodu NKB

Liczba kodowana	Kod NKB
7	0111
0	0000
14	1110
9	1001

Kod prosty BCD

Sposób konstruowania słowa kodowego w kodzie prostym BCD jest następujący:

1. Każdej cyfrze dziesiętnej przyporządkowujemy czterocyfrową liczbę dwójkową (zwaną tetradą) w kodzie NKB (gdybyśmy zamiast słów kodu NKB użyli innego kodu, np. kodu Gray'a, wówczas otrzymalibyśmy kod BCD Gray'a). Przyporządkowanie to przedstawione jest w tabeli 1.3.

Tabela 1.3. Przyporządkowanie cyfr dziesiętnych tetradom NKB

Cyfra dziesiętna	Tetrada NKB	Cyfra dziesiętna	Tetrada NKB
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

2. Słowo kodowe w kodzie prostym BCD, odpowiadające danej liczbie otrzymujemy, zapisując każdą cyfrę tej liczby w postaci czwórki cyfr binarnych, zgodnie z tabelą 1.3.

Przykład

Znaleźć słowa kodu prostego BCD odpowiadające liczbom 463_D i 67_D .

Rozwiązanie

Jeśli zapiszemy każdą cyfrę liczby w postaci tetrady NKB, otrzymamy:

$$463_D = 0100 \ 0110 \ 0011_{BCD}$$

$$67_D = 0110 \ 0111_{BCD}$$

Kod ASCII

Innym przykładem kodu jest kod służący do kodowania tekstów i przesyłania ich pomiędzy urządzeniami cyfrowymi. Nosi on nazwę kodu ASCII (ang. *American Standard Code for Information Interchange*). Koduje on oprócz znaków alfanumerycznych tak zwane znaki sterujące, służące do sterowania transmisją i pracą drukarki lub dalekopisu. Kod ten podamy w postaci tabeli zawierającej kodowane obiekty i odpowiadające im słowa kodowe (tabela 1.4). Jak widać, tabela oprócz znaków alfanumerycznych zawiera tak zwane znaki sterujące, służące do sterowania transmisją i pracą drukarki/dalekopisu. Pełne zestawienie znaków sterujących wraz z ich znaczeniem zawiera tabela 1.5.

Tabela 1.4. Kod ASCII

Numery bitów słowa				8	bit kontroli parzystości							
				7	0	0	0	0	1	1	1	1
				6	0	0	1	1	0	0	1	1
				5	0	1	0	1	0	1	0	1
4	3	2	1									
0	0	0	0		NUL	DLE	SP	0	@	P	'	p
0	0	0	1		SOH	DC1	!	1	A	Q	a	q
0	0	1	0		STX	DC2	”	2	B	R	b	r
0	0	1	1		ETX	DC3	≠	3	C	S	c	s
0	1	0	0		EOT	DC4	\$	4	D	T	d	t
0	1	0	1		ENQ	NAK	%	5	E	U	e	u
0	1	1	0		ACK	SYN	&	6	F	V	f	v
0	1	1	1		BEL	ETB	'	7	G	W	g	w

Numery bitów słowa				8	bit kontroli parzystości							
				7	0	0	0	0	1	1	1	1
				6	0	0	1	1	0	0	1	1
				5	0	1	0	1	0	1	0	1
1	0	0	0	0	BS	CAN	(	8	H	X	h	x
1	0	0	1	1	HT	EM	)	9	I	Y	i	y
1	0	1	0	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	1	VT	ESC	+	;	K	[	k	{
1	1	0	0	0	FF	FS	,	<	L	\	l	
1	1	0	1	1	CR	GS	-	=	M	]	m	}
1	1	1	0	0	SO	RS	.	>	N	↑	n	~
1	1	1	1	1	SI	US	/	?	O	←	o	DEL

Tabela 1.5. Znaki sterujące kodu ASCII

Symbol	Pełna nazwa znaku	Polska nazwa znaku
SOH	Start of heading	Początek nagłówka
STX	Start of text	Początek tekstu
ETX	End of text	Koniec tekstu
EOT	End of transmission	Koniec transmisji
ENQ	Enquiry	Zapytanie
ACK	Acknowledge	Potwierdzenie
DLE	Data line escape	Zmiana interpretacji określonej liczby słów kodu
NAK	Negative acknowledge	Brak potwierdzenia
SYN	Synchronous file	Znak synchronizujący
ETB	End of transmission block	Koniec transmisji bloku
NUL	Null	Znak pusty
BEL	Bell	Dzwonek
SO	Shift out	Wyjście z kodu
SI	Shift in	Powrót do kodu
CAN	Cancel	Anulowanie danych zawartych w bloku
EM	End of medium	Koniec nośnika informacji
SUB	Substitute	Zamiana błędnego znaku

Symbol	Pełna nazwa znaku	Polska nazwa znaku
ESC	Escape	Zmiana kodu
DEL	Delete	Unieważnienie
BS	Backspace	Cofnij o jedną pozycję
HT	Horizontal tabulation	Tabulacja pozioma
LF	Line feed	Zmiana wiersza
VT	Vertical tabulation	Tabulacja pionowa
CR	Carriage return	Powrót karetki
FF	Form feed	Zmiana formatu, arkusza, strony
FS	File separator	Separator zbiorów
GS	Group separator	Separator grup
RS	Record separator	Separator zapisów
US	Unit separator	Separator jednostek
SP	Space	Spacja
DC1	Device control 1	Sterowanie urządzenia 1
DC2	Device control 2	Sterowanie urządzenia 2
DC3	Device control 3	Sterowanie urządzenia 3
DC4	Device control 4	Sterowanie urządzenia 4

Podane przykłady nie wyczerpują oczywiście możliwości kodowania informacji. Prócz kodowania informacji czysto liczbowej czy też tekstów możemy przykładowo kodować (i przetwarzać cyfrowo) obrazy, dźwięk, wielkości fizyczne takie jak temperatura, ciśnienie (odpowiednie czujniki plus przetworniki a/c) oraz wiele innych. Przykłady niektórych kodów (m.in. kodu U2) zostaną podane później wraz z przykładowymi zastosowaniami.

1.3. Informacja cyfrowa

Terminu *informacja cyfrowa* używaliśmy już w poprzednich podrozdziałach, opierając się na intuicji. Tutaj podamy jej definicję a także ustalimy pewną terminologię dotyczącą słów informacji cyfrowej o określonej długości.

Definicja

Słowem cyfrowym (binarnym) nazywamy dowolny ciąg składający się z symboli 0 i/lub 1.

Definicja

Informacją cyfrową nazywamy informację przedstawioną (zakodowaną) w postaci słów cyfrowych.

Z różnych względów, na przykład z powodu stosowania do przesyłania informacji dróg o określonej szerokości zwanych magistralami, pewne określone długości słów mają swoje nazwy. Zestawienie najczęściej spotykanych słów cyfrowych oraz ich nazw angielskich i polskich zawiera tabela 1.6.

Tabela 1.6. Nazwy słów cyfrowych

Długość słowa	Oznaczenie symboliczne	Nazwa angielska	Nazwa polska
1	a_0	binary digit, bit	bit
4	$a_3 \dots a_0$	nibble	tetrada, kęs
8	$a_7 \dots a_0$	byte	bajt
16	$a_{15} \dots a_0$	16-bit word, word	słowo 16-bitowe, słowo
32	$a_{31} \dots a_0$	double word, dword	podwójne słowo, dwusłowo
64	$a_{63} \dots a_0$	quad word, qword	słowo 64-bitowe, czterosłowo

Jeden bit jest jednocześnie jednostką (inaczej mówiąc, najmniejszą porcją) ilości informacji.

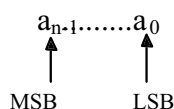
W celu uniknięcia niejednoznaczności przy podawaniu ilości informacji ustalamy, że skrótowe oznaczenie:

1b – oznacza 1 bit

1B – oznacza 1 bajt.

Na przykład 20 MB jest ilością informacji ośmiokrotnie większą niż 20 Mb.

W słowach informacji cyfrowej warto też wyróżnić najstarszą i najmłodszą pozycję. Nazywa się je odpowiednio *najbardziej znaczącym bitem* lub *najstarszym bitem* (ang. *MSB – most significant bit*) oraz *najmniej znaczącym bitem* lub *najmłodszym bitem* (ang. *LSB – least significant bit*). Pozycje obydwu bitów w symbolicznym słowie cyfrowym przedstawia rysunek 1.2.



Rysunek 1.2. Najbardziej i najmniej znaczący bit

Analogicznie w słowie możemy mówić o starszym bajcie lub o młodszym bajcie, czy też w bajcie o starszej lub młodszej tetradzie.

1.4. Działania logiczne i bramki

Jedną z ważnych grup działań wykonywanych przy przetwarzaniu informacji są działania logiczne. Wykonywanie tych działań wiąże się z operowaniem dwoma wartościami logicznymi zwanymi *prawdą* (ang. *true*) i *fałszem* (ang. *false*). Wartości te są w logice pojęciami pierwotnymi, czyli nie są to pojęcia definiowane. Działania logiczne operują na wartościach logicznych i ich wynikiem jest też wartość logiczna (podobnie jak działania arytmetyczne operują na liczbach, dając w wyniku liczbę). W układach cyfrowych wartości logiczne muszą być reprezentowane (jak każdy rodzaj informacji) przez dwa stany elektryczne: wysoki – **H** i niski – **L**. Możemy przyporządkować (czyli zakodować) wartości logicznej **prawda** stan **H**, zaś wartości logicznej **fałsz** stan **L**. Posługujemy się wówczas tak zwaną *logiką prostą*. Przy odwrotnym przyporządkowaniu mamy do czynienia z *logiką odwróconą*. Jak wspomnieliśmy, w technice komputerowej stan **H** jest zwykle zapisywany jako **1**, a stan **L** jako **0**, stąd przy prezentacji działań logicznych będziemy posługiwać się właśnie tymi oznaczeniami.

Przedstawiając działania logiczne, posłużymy się dwoma metodami. Pierwsza z nich to opis słowny. Druga z nich to tak zwana tabela prawdy, która wymaga krótkiego wyjaśnienia. Polega ona na przedstawieniu w tabeli wszystkich możliwych kombinacji argumentów i odpowiadających im wartości logicznych wyniku danego działania. Przypomina więc ona przykładowo określenie działania zwanego w arytmetyce mnożeniem w postaci tabliczki mnożenia.

W technice cyfrowej działania logiczne wykonywane są przez układy cyfrowe zwane *bramkami*. Wyjaśnienie, skąd wziął się termin „bramka”, odkładamy na później. Bramki są podstawowymi układami cyfrowymi, będącymi „cegiełkami”, z których buduje się bardziej skomplikowane układy logiczne. Przy opisie bramek, a później przy opisie wielu innych układów cyfrowych, będziemy posługiwali się tak zwaną

metodą *czarnej skrzynki*. Polega ona na tym, że nie zastanawiamy się, dlaczego układ działa w dany sposób ani jaka jest jego wewnętrzna budowa, a interesuje nas jedynie jego zachowanie zewnętrzne. Inaczej mówiąc, będziemy podawać, jak przy określonych sygnałach wejściowych (pobudzeniach) będzie się zachowywać wyjście układu. Dla niektórych typów układów (na przykład dla mikroprocesora) jest to praktycznie jedyny możliwy sposób opisu (ze względu na stopień komplikacji tych układów). Zobaczmy ponadto, że w przypadku bramek do ich opisu będziemy mogli zastosować tabelę prawdy.

Przedstawiając działania logiczne, będziemy też używać terminu zmiennej logicznej.

Definicja

Zmienną logiczną nazywamy zmienną, która może przyjmować jedną z dwóch wartości logicznych: prawdę lub fałsz

W zapisie stosowanym w układach cyfrowych zmienna taka będzie więc przyjmować wartość 1 bądź 0, przy czym pamiętamy, że w tym wypadku są to zakodowane wartości logiczne.

Zwyczajowo zmienne logiczne oznacza się małymi literami z końca alfabetu. Działania logiczne można przedstawić jako działania na zmiennych logicznych. Po tych krótkich wyjaśnieniach przechodzimy do zdefiniowania podstawowych działań logicznych: iloczynu logicznego, sumy logicznej oraz negacji (zaprzeczenia). Określenie pozostałych działań logicznych (na przykład alternatywy wykluczającej, inaczej funkcji EX-OR, czy też działań złożonych, takich jak funkcja NAND, NOR lub EX-NOR), można znaleźć na przykład w pozycji [22].

1. Iloczyn logiczny – bramka **AND**.

Iloczyn logiczny dwóch zmiennych zapisujemy jako:

$$y = x_1 \wedge x_2$$

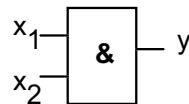
i czytamy „y równa się x_1 i x_2 ”. W układach cyfrowych często zamiast symbolu „ $\wedge$ ” używa się symbolu zwykłego mnożenia, stąd powyższe działanie można zapisać jako:

$$y = x_1 \bullet x_2$$

zaś z kontekstu tego zapisu wynika, czy jest to działanie logiczne, czy arytmetyczne.

Symbol graficzny układu cyfrowego realizującego iloczyn logiczny, czyli symbol bramki AND przedstawia rysunek 1.3, zaś tabela 1.7 jest tabelą prawdy opisującą

zarówno zachowanie się tej bramki, jak i własności dwuargumentowego iloczynu logicznego.

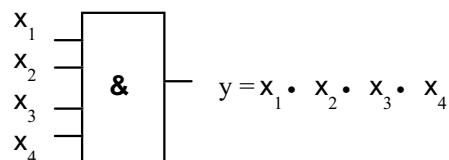


Rysunek 1.3. Symbol bramki AND

Tabela 1.7. Tabela prawdy dwuwejściowej bramki AND

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Z tabeli tej łatwo odczytać, że wartość iloczynu logicznego dwóch zmiennych jest równa 1 (czyli prawdziwie) tylko wtedy, gdy obydwie wartości argumentów wynoszą 1. W pozostałych przypadkach otrzymujemy 0. Można to uogólnić na wiele argumentów, stwierdzając, że iloczyn logiczny jest prawdziwy, gdy wszystkie argumenty tego iloczynu są prawdziwe. Iloczyn wieloargumentowe realizowane są przez bramki wieloargumentowe. Symbol przykładowej czterowejściowej bramki AND oraz zależność sygnału wyjściowego od sygnałów wejściowych przedstawia rysunek 1.4.



Rysunek 1.4. Czterowejściowa bramka AND

2. Suma logiczna – bramka **OR**.

Tabela 1.8 pokazuje wartości dwuargumentowej sumy logicznej w zależności od wartości jej argumentów, zaś rysunek 1.5 przedstawia symbol graficzny dwuwejściowej bramki OR realizującej to działanie.

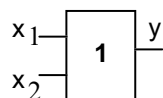
Tabela 1.8. Tabela prawdy dwuwejściowej bramki OR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Sumę logiczną zapisujemy jako:

$$x_1 \vee x_2 = y \quad \text{lub} \quad x_1 + x_2 = y$$

i czytamy jako „ x_1 lub x_2 ”. Podobnie jak w poprzednim punkcie, możemy działanie to uogólnić na wiele argumentów, podając następujące jej określenie. Suma logiczna jest równa zero tylko wtedy, gdy wszystkie argumenty są równe 0. W pozostałych przypadkach wynikiem działania jest 1.



Rysunek 1.5. Symbol bramki OR

3. Negacja – bramka **NOT**.

Operację negacji, czyli zaprzeczenia oznaczamy następująco:

$$y = \sim x \quad \text{lub} \quad y = \bar{x}$$

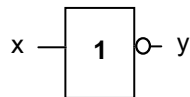
Pierwsze oznaczenie stosowane jest głównie przez matematyków, drugie jest bardzo chętnie stosowane w układach cyfrowych (choć niezbyt chętnie jest ono stosowane przez składających teksty). Firma Intel sygnały zanegowane oznacza jeszcze inaczej, a mianowicie:

$$y = x \#$$

Tabela 1.9. Tabela prawdy bramki NOT

x	y
0	1
1	0

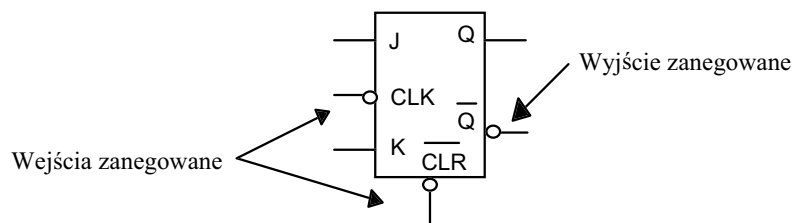
W każdym przypadku czytamy „nie x” lub „nieprawda, że x”.



Rysunek 1.6. Symbol bramki NOT

Określenie negacji jest bardzo proste. Jeżeli wartość argumentu jest równa 0 (fałsz), to w wyniku otrzymujemy 1 (prawda) i odwrotnie. Negacja działa zawsze na jeden argument (choć możemy zanegować na przykład sumę). Stąd bramka NOT jest zawsze jednoweściowa. Jej symbol oraz tabelę prawdy przedstawia rysunek 1.6 i tabela 1.9.

Symbol negacji wymaga krótkiego komentarza. Ponieważ sygnały logiczne mogą być negowane zarówno na wejściach, jak i na wyjściach układów, przyjęto, że operację negacji oznacza w symbolu bramki NOT kółeczko. Dlatego wejścia bądź wyjścia sygnału zanegowanego w układach cyfrowych są oznaczane tak, jak to pokazano na rysunku 1.7.

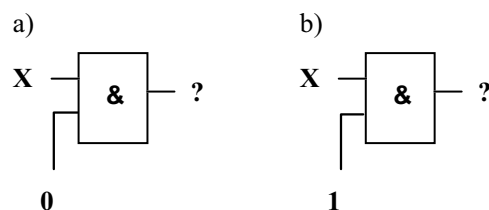


Rysunek 1.7. Sposób oznaczania wejść i wyjść zanegowanych

Na koniec niniejszego podpunktu spróbujemy na przykładzie uzasadnić nazwę układu „bramka”.

Przykład

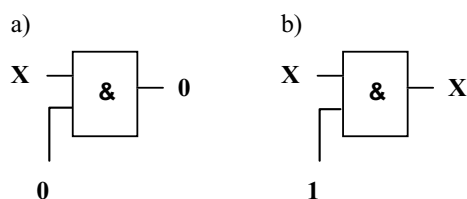
Określić, jaki będzie stan na wyjściu bramki pokazanej na rysunku 1.8, gdy na jedno z jej wejść podajemy sygnał binarny X, zaś na drugie wejście: a) stan 0; b) stan 1.



Rysunek 1.8. Rysunek do przykładu wyjaśniającego pochodzenie terminu „bramka”

Rozwiązanie

Zgodnie z tabelą prawdy (lub definicją iloczynu), jeżeli choć na jednym wejściu bramki jest **0**, to na wyjściu jest także zero, niezależnie od stanu pozostałych wejść. W przypadku gdy na jednym z wejść bramki dwuwejściowej jest sygnał **1**, wówczas stan wyjścia tej bramki zależy od stanu drugiego wejścia. Jeżeli na tym wejściu jest **0** to i na wyjściu **0** i to samo dla stanu **1**. Możemy więc stwierdzić, że na wyjściu tej bramki występuje sygnał X. Tłumaczy to wynik pokazany na rysunku 1.9.



Rysunek 1.9. Rozwiązanie przykładu ze strony 13

Traktując stan drugiego wejścia jako pewien sygnał sterujący, widzimy, że gdy ma on wartość **1**, wówczas sygnał z drugiego wejścia jest przenoszony na wyjście a gdy sygnał sterujący ma wartość **0**, sygnał X jest blokowany, czyli... *bramkowany*. Stąd nazwa układu.

1.5. Podział układów logicznych

W zależności od przyjętego kryterium, możemy wyróżnić kilka sposobów podziału układów logicznych. Poniżej podamy dwa z nich, związane ze sposobem funkcjonowania układów logicznych. Układy logiczne podzielimy na układy kombinacyjne i sekwencyjne oraz na układy asynchroniczne i synchroniczne.

1.5.1. Układy kombinacyjne i sekwencyjne

Definicja

Układem kombinacyjnym nazywamy taki układ cyfrowy, w którym stan wejść jednoznacznie określa stan wyjść układu.

Oznacza to, że aby określić stan na wyjściach takiego układu, nie potrzebujemy żadnej dodatkowej informacji (poza stanem wejść i rodzajem układu). Najprostszym przykładem układów kombinacyjnych są bramki. Jedną z cech układów kombinacyjnych jest możliwość przedstawienia ich działania (opisu) w postaci tabeli prawdy. Jest to oczywiste, gdyż tabela prawdy podaje właśnie zależność sygnałów wyjściowych od

wejściowych. Inną cechą układów kombinacyjnych jest możliwość ich realizacji poprzez proste połączenie odpowiedniej ilości i rodzaju bramek, bez sprzężeń zwrotnych (czyli prowadzenia sygnałów wstecz, od wyjścia do wejścia). Inne własności będą miały układy sekwencyjne.

Definicja

Układem sekwencyjnym nazywamy układ cyfrowy, w którym stan wyjść zależy od stanu wejść oraz od poprzednich stanów układu.

Oznacza to, że układy sekwencyjne są układami z pamięcią. Klasycznym przykładem może być tu licznik. Znajomość stanu jego wejścia zliczającego – „pojawił się kolejny impuls do zliczenia”, nie pozwala jeszcze określić, jaka liczba zliczonych impulsów pojawiła się na jego wyjściu. Do określenia tej wielkości potrzebna jest nam znajomość liczby impulsów, które wcześniej zliczył licznik (i którą musiał pamiętać). Najprostszymi układami sekwencyjnymi są przerzutniki, których definicję i przykłady podamy w następnym podrozdziale.

1.5.2. Układy asynchroniczne i synchroniczne

Podamy teraz definicje dwóch kolejnych klas układów cyfrowych związanych z podziałem na układy asynchroniczne i synchroniczne.

Definicja

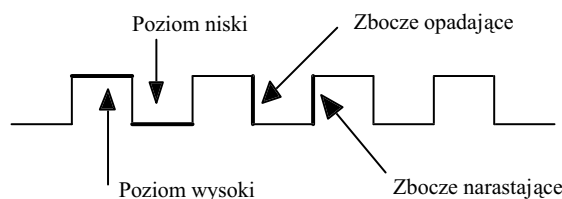
Układem asynchronicznym nazywamy taki układ cyfrowy, dla którego w dowolnym momencie jego działania stan wejść oddziałuje na stan wyjść.

Dla układów tych możemy zatem stwierdzić, że ich własności nie zależą od żadnego przebiegu czasowego.

Definicja

Układem synchronicznym nazywamy taki układ cyfrowy, dla którego stan wejść wpływa na stan wyjść jedynie w pewnych określonych odcinkach czasu pracy układu zwanych *czasem czynnym*, natomiast w pozostałych odcinkach czasu zwanych *czasem martwym* stan wejść nie wpływa na stan wyjść. Odcinki czasu czynnego i martwego wyznaczane są przez podanie specjalnego przebiegu zwanego przebiegiem zegarowym lub taktującym na wejście zwane wejściem zegarowym lub taktującym.

Własności układów synchronicznych zależą więc od zmian przebiegu zegarowego w funkcji czasu. Przebieg ten dla układów cyfrowych jest z reguły tak zwanym przebiegiem prostokątnym, czyli przyjmującym dwa poziomy, wysoki i niski, rozdzielone momentami zmian zwanymi zboczami: narastającym (zmiana od stanu niskiego do wysokiego) i opadającym (zmiana od stanu wysokiego do niskiego). Przebieg tego typu pokazany jest na rysunku 1.10.



Rysunek 1.10. Cyfrowy przebieg zegarowy

W zależności od jednego z czterech wyróżnionych fragmentów tego przebiegu używanych do wyznaczania czasu czynnego układu, możemy mówić o układach synchronicznych reagujących na poziom wysoki bądź niski lub na zbocze narastające bądź opadające. Oznaczenia każdego typu wejścia zegarowego podaje tabela 1.10.

Dla układów synchronicznych mamy możliwość dokładnego wyznaczenia momentu wpływu sygnałów wejściowych na stan wyjść układu. Ma to duże znaczenie praktyczne, gdyż pozwala na przykład unikać pewnych zakłóceń czy też wyznaczać moment reakcji układu dopiero wtedy, gdy sygnały wejściowe znajdują się w stanie ustalonym. Bardzo wiele układów opisywanych w dalszej części książki będzie układami synchronicznymi.

Tabela 1.10. Oznaczenia wejść zegarowych układów cyfrowych

Rodzaj wejścia zegarowego	Symbol graficzny
układ reaguje na poziom wysoki	— CLK
układ reaguje na poziom niski	—○ CLK
układ reaguje na zbocze narastające	—▷ CLK
układ reaguje na zbocze opadające	—○▷ CLK

1.6. Przerzutniki

Przerzutniki są najprostszymi układami sekwencyjnymi, czyli najprostszymi układami z pamięcią. Wraz z bramkami pełnią one rolę „cegiełek”, z których buduje się bardziej skomplikowane układy cyfrowe.

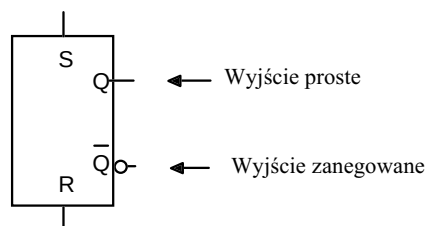
Definicja

Przerzutnikiem nazywamy układ cyfrowy pozwalający zapamiętać jeden bit informacji.

W zależności od sposobu wprowadzania informacji do przerzutnika wyróżniamy kilka podstawowych rodzajów przerzutników oznaczanych symbolicznie skrótami literowymi: RS, JK, D, T. Niektóre z nich mogą być układami asynchronicznymi, jednakże w większości są to układy synchroniczne. Poniżej opiszemy dla przykładu przerzutniki: asynchroniczny RS i synchroniczny D z wejściem reagującym na poziom (czyli tak zwany przerzutnik typu *latch* – zatrask). Obydwa przerzutniki opiszemy, używając metody czarnej skrzynki.

1.6.1. Asynchroniczny przerzutnik RS

Symbol graficzny takiego przerzutnika przedstawia rysunek 1.11, zaś jego działanie opisuje tabela 1.11. Podaje ona, jaki stan zostanie wpisany do przerzutnika w zależności od różnych kombinacji wartości sygnałów wejściowych R i S. Oznaczenie wejścia „S” pochodzi od angielskiego słowa *set* czyli „ustaw” i oznacza wpis jedynki. „R” pochodzi od *reset* i oznacza zerowanie, czyli wpis zera.



Rysunek 1.11. Symbol przerzutnika RS

Jak widać na rysunku, prezentowany przerzutnik ma dwa wyjścia, *wyjście proste* i *wyjście zanegowane*. Stan wyjścia prostego powinien być zawsze przeciwny do stanu wyjścia zanegowanego.

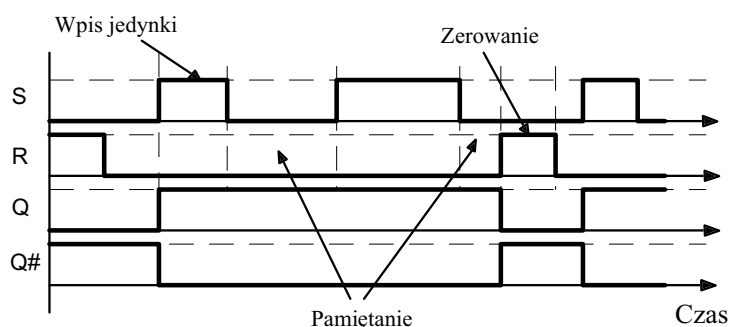
Tabela 1.11. Tabela charakterystyczna przerzutnik RS

R	S	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	—

W tabeli Q_n oznacza stan na wyjściu przed zmianą sygnałów sterujących, natomiast Q_{n+1} stan po ich zmianie. Pozioma kreska oznacza tak zwany stan logicznie zabroniony.

Interpretacja tabeli jest następująca. Jeżeli na obydwu wejściach układu są zera, wówczas nie żądamy od przerzutnika wykonania żadnej operacji i jest on wówczas w stanie *pamiętania*. Oznacza to, że stan przerzutnika nie zmienia się, czyli że $Q_{n+1} = Q_n$. Gdy podajemy stan **1** na wejście S, żądamy wpisu jedynki i wówczas $Q_{n+1} = 1$. Podanie stany **1** na wejście R oznacza zerowanie przerzutnika, czyli $Q_{n+1} = 0$. Podanie jedynek na obydwa wejścia przerzutnika jest żądaniem operacji niewykonalnej, a mianowicie jednoczesnego wpisu zera i jedynki, co jest sprzecznością (także logiczną). Rzeczywiste przerzutniki będą na taki stan reagować niezgodnie z naszymi wymaganiami (określonymi w tabeli ich działania). Przykładowo na wyjściach Q i $\bar{Q}$ pojawi się ten sam stan, co jest niezgodne z definicją tych wyjść.

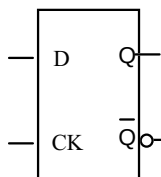
Działanie przerzutnika można też przedstawić przy pomocy tak zwanych diagramów czasowych, czyli zmian wartości sygnałów na jego wyjściach w czasie, w zależności od zmieniających się w czasie sygnałów wejściowych. Przykład takich diagramów dla asynchronicznego przerzutnika RS przedstawia rysunek 1.12.



Rysunek 1.12. Przebiegi czasowe dla przerzutnika synchronicznego RS

1.6.2. Przerzutnik D typu „latch”

Przerzutnik ten jest przerzutnikiem synchronicznym reagującym na poziom (niski lub wysoki, w zależności od rodzaju wejścia zegarowego). Symbol i tabela opisująca jego działanie przedstawione są na rysunku 1.13 i w tabeli 1.12.



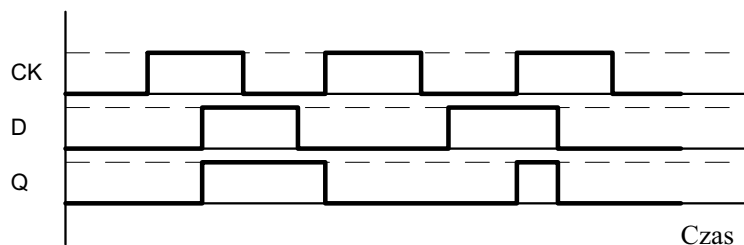
Rysunek 1.13. Symbol przerzutnika D

Należy pamiętać, że przerzutnik ten jest układem synchronicznym, a zatem reakcje zgodne z tabelą 1.12 będą zachodzić tylko w jego czasie czynnym. W czasie martwym stan przerzutnika nie będzie ulegał zmianie, czyli będzie on w stanie pamiętania.

Tabela 1.12. Tabela charakterystyczna przerzutnika D

D	Q_{n+1}
0	0
1	1

Zawartość tabeli oznacza, że jeżeli na wejściu zegarowym jest stan **1** (czas czynny przerzutnika), to gdy na wejściu D jest stan **1**, jest on przepisywany na wyjście (podobnie dla stanu **0**). Inaczej mówiąc, w stanie czynnym przerzutnik typu *latch* powtarza na wyjściu kształt przebiegu z wejścia D, natomiast w momencie przejścia sygnału zegarowego ze stanu aktywnego w nieaktywny (zbrocze opadające) stan z wejścia D jest zapamiętywany i nie zmienia się aż do kolejnego zbocza narastającego. Przykładowe diagramy czasowe opisujące działanie tego przerzutnika pokazane są na rysunku 1.14.



Rysunek 1.14. Przebiegi czasowe dla przerzutnika typu *latch*

Przerzutniki tego typu są używane między innymi do budowy rejestrów typu *latch*, pełniących ważne funkcje w układach techniki komputerowej.