mgr inż. Stanisław Gardynik

Podstawy techniki mikroprocesorowej (1)

„Narzędzia, których próbujemy używać, język lub notacja, których używamy dla wyrażenia lub rejestrowania naszych myśli, są głównymi czynnikami określającymi to, co w ogóle możemy myśleć lub wyrazić!"
E.W. Dijkstra

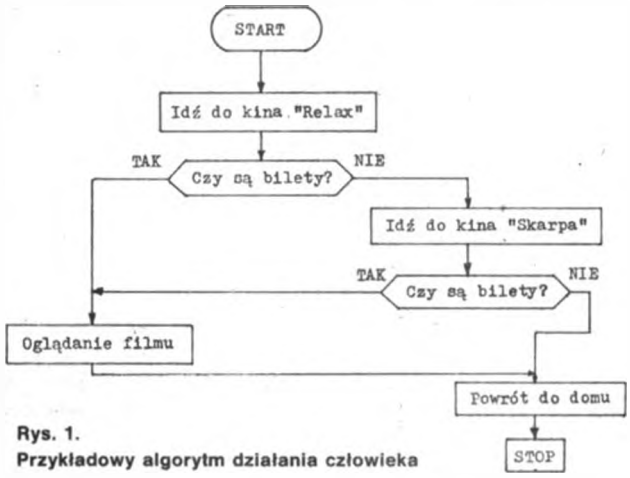
Ta myśl znanego informatyka wyraża pogląd, że narzędzia i środki jakimi dysponujemy do wykonania postawionych przed nami zadań, determinują możliwość ich realizacji. Dlatego też rozpoczynając cykl artykułów o podstawach techniki mikroprocesorowej, przeznaczony dla młodych czytelników, mamy nadzieję, że umożliwi on poznanie sposobu „myślenia" mikroprocesora i jego możliwości, że wyrobi pogląd na to, co można i warto zrealizować w tej nowoczesnej technice. Sądzimy, że nietypowa, czasami zabawna forma tego wykładu pozwoli bezboleśnie wejść w nową przygodę z mikroelektroniką. Przedstawiony materiał jest częścią cyklu książeczek napisanych przez Pana S. Gardynika, dostępnych w jego firmie, ul. Olszowa 68, 05-090 Raszyn.

Od czego zacząć — od pojęć podstawowych.

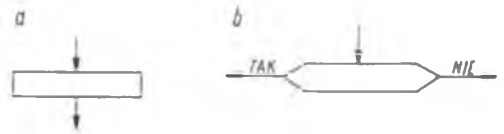
Program komputerowy

Program, to napisany przez człowieka ciąg rozkazów dla komputera. Komputer wykonuje kolejno te rozkazy realizując ściśle określony algorytm działania wymyślony przez człowieka. Zobaczmy na przykładzie czym jest algorytm działania. Załóżmy, że chcemy pójść do kina na film pt. „Seksmisja". Z gazety codziennej dowiadujemy się, że film ten jest wyświetlany tylko w dwóch kinach: „Relax" i „Skarpa". Nasz sposób — algorytm działania może wyglądać tak, jak na rys. 1.

W tzw. schemacie blokowym algorytmu możemy wyróżnić dwa typy bloków: blok funkcjonalny i blok warunkowy, których umowna reprezentacja graficzna jest przedstawiona na rys. 2. Wykonując taki algorytm stajemy się podobni do komputera. Różnica jest jednak zasadnicza. Człowiek może modyfikować powyższy algorytm w trakcie jego wykonywania (np.



Rys. 1.
Przykładowy algorytm działania człowieka

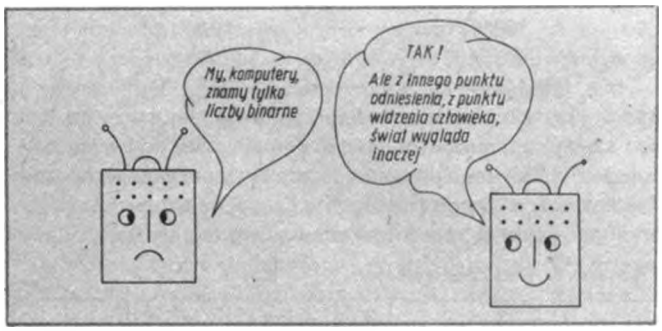


Rys. 2. Reprezentacja graficzna bloków
a — funkcjonalnego, b — warunkowego

w przypadku braku biletów pójść do teatru), komputer natomiast wykonuje program ściśle wg algorytmu, który wymyślił człowiek.

Przeciętny człowiek obserwując dzisiejsze komputery jest zafascynowany ich możliwościami. Widzi, że potrafią one pisać, malować, rysować, sterować fabryką bez ludzi itp. Nie wie natomiast że...

Więc, aby „zrozumieć" język komputera my ludzie musimy poznać...

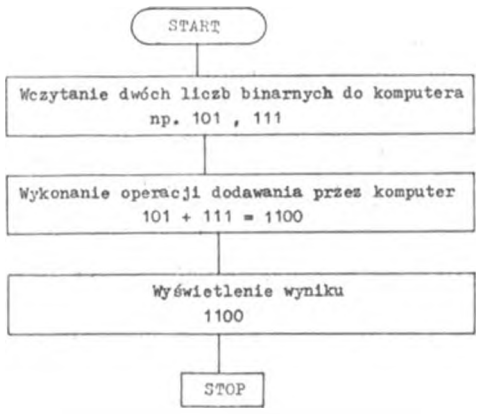


Liczby binarne

Gdyby nasi przodkowie nie wymyślali cyfr 2, 3, 4, 5, 6, 7, 8, 9, a znali tylko cyfry 0 i 1 z pewnością znakomicie posługiwalibyśmy się liczbami binarnymi i mielibyśmy naturalny, wspólny z komputerami język. Komunikacja komputera z człowiekiem pragnącym wykonać np. dodawanie dwóch liczb binarnych wyglądałaby jak na rys. 3.

Oczywiście, dla każdego człowieka posługującego się na codzień liczbami dziesiętnymi taka komunikacja z komputerem byłaby niezwykle uciążliwa. W rzeczywistości, każdy większy komputer ma specjalne programy przetwarzające dane wejściowe i wyjściowe tak, aby człowiek mógł komunikować się z nim w wygodnym dla niego zapisie dziesiętnym. Przykładowy algorytm takiej komunikacji przedstawiono na rys. 4.

Jak widać, w tym przypadku człowiek wprowadza do komputera dane wyjściowe w postaci wygodnych dla niego liczb



Rys. 3. Algorytm dodawania liczb binarnych

dziesiętnych i otrzymuje wynik będący również liczbą dziesiętną.

Zapis ogólny liczby binarnej jest następujący:

$$b_n b_{n-1} \ldots b_2 b_1 b_0 = b_n \cdot 2^n + b_{n-1} \cdot 2^{n-1} + \ldots + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$$

przy czym cyfry binarne, zwane często bitami b_n, b_{n-1}, ...b_2, b_1, b_0 mogą przyjmować wartości wyłącznie 0 albo 1.

Porównajmy powyższy zapis z zapisem ogólnym znanej nam doskonale liczby dziesiętnej:

$$a_n a_{n-1} \ldots a_2 a_1 a_0 = a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + \ldots + a_2 \cdot 10^2 + a_1 \cdot 10^1 + a_0 \cdot 10^0$$

w którym cyfry dziesiętne a_n, a_{n-1}...a_2, a_1, a_0 mogą przyjmować jedną z wartości {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.

Analogie powyższych dwóch zapisów są chyba dla każdego oczywiste. Przykłady liczb dziesiętnych:

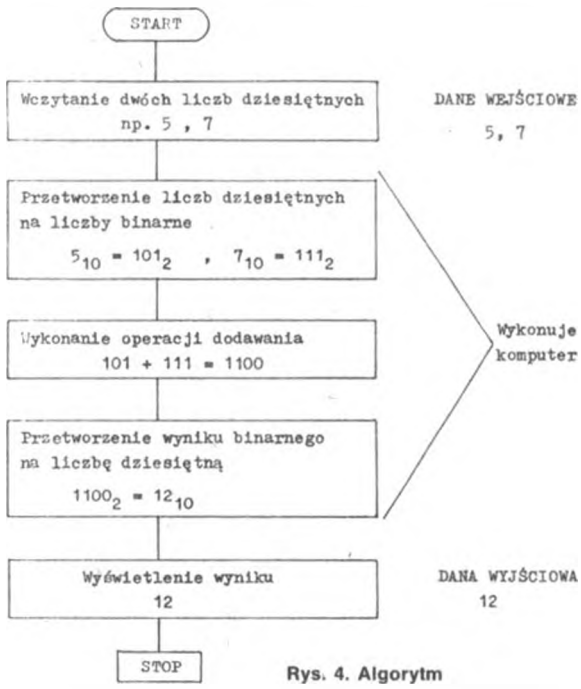
$a_3 a_2 a_1 a_0 = a_3 \cdot 10^3 + a_2 \cdot 10^2 + a_1 \cdot 10^1 + a_0 \cdot 10^0$

$0\ 0\ 2\ 5 = \mathbf{0} \cdot 10^3 + \mathbf{0} \cdot 10^2 + \mathbf{2} \cdot 10^1 + \mathbf{5} \cdot 10^0$

$0\ 2\ 0\ 1 = \mathbf{0} \cdot 10^3 + \mathbf{2} \cdot 10^2 + \mathbf{0} \cdot 10^1 + \mathbf{1} \cdot 10^0$

$1\ 9\ 8\ 5 = \mathbf{1} \cdot 10^3 + \mathbf{9} \cdot 10^2 + \mathbf{8} \cdot 10^1 + \mathbf{5} \cdot 10^0$

gdzie: $10^0 = 1$, $10^1 = 10$, $10^2 = 100$ ...



Rys. 4. Algorytm dodawania liczb dziesiętnych

Przykłady liczb binarnych:

$b_3b_2b_1b_0 = b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$

$0\ 0\ 1\ 0 = \mathbf{0} \cdot 2^3 + \mathbf{0} \cdot 2^2 + \mathbf{1} \cdot 2^1 + \mathbf{0} \cdot 2^0$

$0\ 1\ 0\ 1 = \mathbf{0} \cdot 2^3 + \mathbf{1} \cdot 2^2 + \mathbf{0} \cdot 2^1 + \mathbf{1} \cdot 2^0$

$1\ 1\ 0\ 1 = \mathbf{1} \cdot 2^3 + \mathbf{1} \cdot 2^2 + \mathbf{0} \cdot 2^1 + \mathbf{1} \cdot 2^0$

gdzie: $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$ ...

Liczbę binarną 1101 należy czytać „jeden, jeden, zero, jeden". Każdą liczbę zapisaną w postaci binarnej można przedstawić w zapisie dziesiętnym i odwrotnie.

Zamiana liczby binarnej na liczbę dziesiętną

Zamieńmy liczbę binarną 1011 na liczbę dziesiętną

$1011 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$

czyli:

$$\begin{array}{rl} & 1 \cdot 2^0 = 1 \cdot 1 = 1 \\ + & 1 \cdot 2^1 = 1 \cdot 2 = 2 \\ + & 0 \cdot 2^2 = 0 \cdot 4 = 0 \\ + & 1 \cdot 2^3 = 1 \cdot 8 = 8 \\ \hline & \text{dziesiętnie} \quad 11 \end{array}$$

Jak widać, liczbie binarnej 1011 odpowiada liczba dziesiętna 11 $\qquad 1011_2 = 11_{10}$

W celu zamiany dowolnej liczby binarnej na liczbę dziesiętną należy skorzystać z zapisu ogólnego liczby binarnej.

Zróbmy dwa inne przykłady:

$$\begin{array}{rl} & b_7b_6b_5b_4b_3b_2b_1b_0 \\ & 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1 \\ & 1 \cdot 1 = 1 \quad (b_0) \\ & 0 \cdot 2 = 0 \\ & 1 \cdot 4 = 4 \\ & 1 \cdot 8 = 8 \\ & 0 \cdot 16 = 0 \\ & 1 \cdot 32 = 32 \\ & 0 \cdot 64 = 0 \\ + & 0 \cdot 128 = 0 \quad (b_7) \\ \hline & \text{dziesiętnie} \quad 45 \\ & 00101101_2 = 45_{10} \end{array}$$

$$\begin{array}{rl} & b_7b_6b_5b_4b_3b_2b_1b_0 \\ & 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ & 1 \cdot 1 = 1 \quad (b_0) \\ & 1 \cdot 2 = 2 \\ & 1 \cdot 4 = 4 \\ & 1 \cdot 8 = 8 \\ & 1 \cdot 16 = 16 \\ & 1 \cdot 32 = 32 \\ & 1 \cdot 64 = 64 \\ + & 1 \cdot 128 = 128 \quad (b_7) \\ \hline & \text{dziesiętnie} \quad 225 \\ & 11111111_2 = 255_{10} \end{array}$$

Z ostatniego przykładu wynika, że za pomocą 8 bitów można zapisać $2^8 = 256$ różnych liczb binarnych (0, 1, 2, 3, ...255). Ogólnie: za pomocą n bitów można zapisać 2^n różnych liczb binarnych.

Warto zapamiętać wagi 8-bitowej liczby binarnej podane w tablicy 1.

Tablica 1 **Wagi poszczególnych bitów liczby binarnej 8-bitowej**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
128	64	32	16	8	4	2	1

Pamiętanie powyższych wag umożliwia stosunkowo szybką zamianę liczby binarnej na dziesiętną.

Przykłady

Liczba binarna

$b_7b_6b_5b_4b_3b_2b_1b_0$

$0\ 1\ 0\ 0\ 1\ 0\ 1\ 0 \longrightarrow 64 + 8 + 2 = 74$

$1\ 0\ 0\ 1\ 0\ 0\ 0\ 1 \longrightarrow 128 + 16 + 1 = 145$

$0\ 0\ 0\ 0\ 0\ 1\ 1\ 1 \longrightarrow 4 + 2 + 1 = 7$

Dla wprawy warto zamienić na liczby dziesiętne następujące liczby binarne:

10101011, 01011111, 11000011, 11101010

Zamiana liczby dziesiętnej na liczbę binarną

Zamieńmy liczbę dziesiętną 11 na liczbę binarną

$11 : 2 = 5 + \frac{1}{2}$ stąd $b_0 = 1$

$5 : 2 = 2 + \frac{1}{2}$ stąd $b_1 = 1$

$2 : 2 = 1 + \frac{0}{2}$ stąd $b_2 = 0$

$1 : 2 = 0 + \frac{1}{2}$ stąd $b_3 = 1$

Liczba dziesiętna 11 odpowiada liczbie binarnej 1011

$11_{10} = 1011_2$

Jak widać, liczbę dziesiętną należy stale dzielić przez 2 zapisując wartość reszty z dzielenia dopóki nie uzyskamy zera.

Inne przykłady:

$$\begin{array}{rcl|l} 44 & : 2 = & 22 & 0 \quad (b_0) \\ 22 & : 2 = & 11 & 0 \\ 11 & : 2 = & 5 & 1 \\ 5 & : 2 = & 2 & 1 \\ 2 & : 2 = & 1 & 0 \\ 1 & : 2 = & 0 & 1 \quad (b_6) \end{array} \qquad \begin{array}{rcr|l} 255 & : 2 = & 127 & 1 \quad (b_0) \\ 127 & : 2 = & 63 & 1 \\ 63 & : 2 = & 31 & 1 \\ 31 & : 2 = & 15 & 1 \\ 15 & : 2 = & 7 & 1 \\ 7 & : 2 = & 3 & 1 \\ 3 & : 2 = & 1 & 1 \\ 1 & : 2 = & 0 & 1 \quad (b_7) \end{array}$$

$44_{10} = 101100_2 \qquad 255_{10} = 11111111_2$

Proponujemy jako ćwiczenie zamienić na liczby binarne następujące liczby dziesiętne: 27, 100, 121, 245

Zapis szesnastkowy liczby binarnej

Zapis szesnastkowy liczby binarnej jest używany w celu łatwego zapamiętywania długich liczb binarnych. Umożliwia on człowiekowi sprawne manipulowanie liczbami binarnymi. Porównajmy:

Liczba binarna		Zapis szesnastkowy
10011010	$\longrightarrow$	9A
00101001	$\longrightarrow$	29
11111111	$\longrightarrow$	FF

Każdą liczbę binarną można (przy niewielkiej wprawie) bardzo szybko zapisać w postaci szesnastkowej i odwrotnie. W celu przedstawienia dowolnej liczby binarnej w zapisie szesnastkowym należy podzielić ją na części po cztery bity licząc od końca, a następnie je zakodować wg tablicy 2.

Tablica 2. **Kod szesnastkowy**

Zapis binarny	0000	0001	0010	0011	0100	0101	0110	0111
Cyfra szesnastkowa	0	1	2	3	4	5	6	7

Zapis binarny	1000	1001	1010	1011	1100	1101	1110	1111
Cyfra szesnastkowa	8	9	A	B	C	D	E	F

Jak widać, w zapisie szesnastkowym cztery cyfry binarne kodowane są za pomocą jednej cyfry szesnastkowej.

Przykłady:

$\underbrace{101}_{5}\ \underbrace{1010}_{A}\ \underbrace{1100}_{C} = 5AC_{16} \qquad \underbrace{1111}_{F}\ \underbrace{1000}_{8}\ \underbrace{0000}_{0}\ \underbrace{1011}_{B} = F80B_{16}$

W literaturze fachowej liczby szesnastkowe zwane są czasami liczbami heksadecymalnymi.

Jednoznaczność zapisu

Zapis: 1011 nie jest jednoznaczny, gdyż nie wiadomo, czy jest to liczba binarna, dziesiętna czy też szesnastkowa. W praktyce programowania mikroprocesorów stosuje się następującą notację:

Liczba dziesiętna	Liczba binarna	Liczba szesnastkowa
1011	1011B	1011H
1985	10101010B	25FAH

Widać, że o typie liczby decyduje ostatnia litera.

B — oznacza, że mamy do czynienia z liczbą binarną (ang. binary)

H — oznacza, że mamy do czynienia z liczbą szesnastkową (ang. hexadecimal)

Brak litery — oznacza, że mamy do czynienia z liczbą dziesiętną.

Czasami stosuje się literę D (ang. decimal).

W języku asemblera, który poznamy w dalszej części elementarza obowiązuje następująca zasada:

Jeżeli liczba szesnastkowa rozpoczyna się od cyfry będącej literą, to należy ją poprzedzić cyfrą 0, np. 0F80BH, 0A0H itp.

Cd. w następnym numerze

mgr inż. Stanisław Gardynik

Podstawy techniki mikroprocesorowej (2)

Dodawanie liczb binarnych

Operacje arytmetyczne na liczbach binarnych są analogiczne do znanych nam dobrze operacji na liczbach dziesiętnych. Wiemy, że przy dodawaniu dwóch cyfr dziesiętnych przeniesienie wystąpi wtedy i tylko wtedy, gdy ich suma jest większa od 9.

$$\begin{array}{r} 3 \\ +\ 5 \\ \hline 8 \end{array} \qquad \begin{array}{r} 2 \\ +\ 7 \\ \hline 9 \end{array} \qquad \begin{array}{r} 3 \\ +\ 7 \\ \hline (1)0 \end{array} \qquad \begin{array}{r} 8 \\ +\ 7 \\ \hline (1)5 \end{array}$$

przy czym: (1) — przeniesienie.

Podobnie przy dodawaniu dwóch liczb binarnych przeniesienie wystąpi wtedy i tylko wtedy, gdy ich suma jest większa od 1.

$$\begin{array}{r} 0 \\ +\ 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ +\ 1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ 0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ 1 \\ \hline (1)0 \end{array}$$

przy czym: (1) — przeniesienie.

Kilka przykładów dodawania dwóch liczb binarnych przedstawiono w tablicy 1. Dla sprawdzenia poprawności wyniku wykonano to samo dodawanie na liczbach dziesiętnych.

W praktyce programowania mikroprocesorów umiejętność szybkiej zamiany liczby binarnej na szesnatkową i odwrotnie jest bardzo pożądana. Znając reguły dodawania liczb binarnych spróbujmy nauczyć się „na pamięć" tablicy 2.

Z powyższego widać, że jeżeli zapamiętamy binarne reprezentacje zaledwie 4 cyfr szesnastkowych, tj. 0, 4, 8, C, łatwo utworzymy binarną reprezentację dowolnej cyfry szesnastkowej dodając w pamięci odpowiednią liczbę jedynek.

Przykład: 4 BH = 01001011B

Z cyfrą 4 nie mieliśmy oczywiście żadnych kłopotów, natomiast cyfrę B utworzyliśmy dodając w pamięci trzy jedynki do cyfry 8. Zauważmy, że zamiast uczyć się na pamięć 16 różnych kombinacji zer i jedynek uczymy się tylko 4 tworząc pozostałe

Tablica 1. **Przykłady dodawania liczb binarnych**

Binarnie	Dziesiętnie	Sprawdzenie
101 + 111 1100	5 + 7 12	1100B = 8 + 4 = 12
10011 + 00011 10110	19 + 3 22	10110B = 16 + 4 + 2 = 22
01111111 + 00000001 10000000	127 + 1 128	10000000B = 128
00101101 + 01010110 10000011	45 + 86 131	10000011B = 128 + 2 + 1 = 131

Tablica 2. **Kod szesnastkowy**

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

wystarczająco szybko i łatwo. Zakładając, że nasz mózg ma ograniczoną pojemność, powyższy sposób uczenia się umożliwia zaoszczędzenie całkiem pokaźnej liczby naszych szarych komórek, do których zamiast bezsensownych danych możemy wpisać jakiś program „logicznego myślenia".

Jeżeli chcesz zostać dobrym mikroelektronikiem, to zmniejszaj bazę danych w swoim mózgu rozwijając „Struktury logicznego myślenia".

Poćwiczmy teraz zamianę liczb szesnastkowych na liczby binarne i odwrotnie korzystając z powyższej „teorii".
Zamieńmy w tym celu liczby szesnastkowe na liczby binarne: C840H, A501H, F946H, D398H, 5CABH, 1234H

Odejmowanie liczb binarnych

Reguły odejmowania dwóch cyfr binarnych są następujące:

```
  0     1     1    (1)0
- 0   - 0   - 1    - 1
-----  -----  -----  -----
  0     1     0     1
```

przy czym: (1) — pożyczka
Porównajmy:

Odejmowanie liczb dziesiętnych

```
  1004
- 0009
-------
  0995
```

Odejmowanie liczb binarnych

```
  1000
- 0001
-------
  0111
```

Kilka przykładów odejmowania dwóch liczb binarnych przedstawiono w tablicy 3. Dla sprawdzenia poprawności wyniku wykonano to samo odejmowanie na liczbach dziesiętnych.

Tablica 3. **Przykłady odejmowania liczb binarnych**

Binarnie	Dziesiętnie	Sprawdzenie
1001 − 0101 0100	9 − 5 4	0100B = 4
10101100 − 01100111 01000101	172 − 103 69	01000101B = 64 + 4 + 1 = 69
10001111 − 00011111 01110000	143 − 31 112	01110000B = 64 + 32 + 16 = 112

Dla wprawy można zamienić poniższe liczby szesnastkowe na liczby binarne, a następnie wykonać operacje dodawania i odejmowania. Wyniki obliczeń zamienić na liczby szesnastkowe:

1A	AC	33	AA	BA
(−) + 12	(−) + 2F	(−) + 1C	(−) + 55	(−) + 37
a.	b.	c.	d.	e.

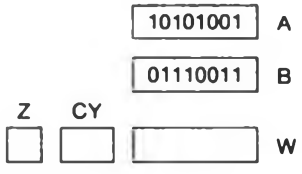
Następnie proponujemy wykonać poprzednie ćwiczenie dodając i odejmując... szesnastkowo.
Najpopularniejsze mikroprocesory 8-bitowe (Z80, 6502, 8085) potrafią wykonywać tylko najprostsze operacje arytmetyczne i logiczne. Pojęcie „mikroprocesorów 8-bitowych" wynika z faktu, że mikroprocesory te operują 8-bitowymi liczbami binarnymi zwanymi najczęściej bajtami.

BAJT (ang. byte) — 8-bitowa liczba binarna.

Przypatrzmy się bliżej jak One to robią.

Mikroprocesorowa realizacja operacji arytmetycznych

Wyobraźmy sobie, że mamy do dyspozycji 5 tablic szkolnych, kredę i ścierkę.



Na większych tablicach o nazwach A, B, W możemy zapisywać tylko i wyłącznie 8-bitowe liczby binarne, zaś na małych tabliczkach CY i Z zmieści się nam tylko jedna cyfra binarna.
W dalszej części niniejszego rozdziału będziemy wykonywać operacje arytmetyczne i logiczne na przykładowych zawartościach tablic A i B. Małe tabliczki CY i Z będą pełniły funkcję wskaźników sygnalizacyjnych (ang. Flags) ustawianych w czasie wykonywania operacji arytmetycznych lub logicznych.
W poprzednich rozdziałach poznaliśmy już reguły dodawania i odejmowania dwóch liczb binarnych. Umiemy zatem wykonać operacje typu:
W = A + B, W = A − B
przy czym: W = A + B oznacza:
Dodaj zawartość tablic A i B wpisując wynik do tablicy o nazwie W.
Reguły ustawiania wskaźników sygnalizacyjnych dla operacji dodawania są następujące:

CY = 1 wtedy i tylko wtedy, gdy wynik dodawania jest liczbą 9-bitową (wystąpi przeniesienie). W przeciwnym przypadku CY = 0

Z = 1 wtedy i tylko wtedy, gdy wynik operacji dodawania jest równy zeru (W = 0). W przeciwnym przypadku Z = 0.

W literaturze fachowej powyższe wskaźniki mają nazwy:
CY — wskaźnik przeniesienia (ang. Carry)
Z — wskaźnik zera (ang. Zero)
Przykłady: W = A + B

```
    (1) 10101001  A          (1) 11111111  A
  +     01111100  B        +     00000001  B
  ---------------------    ---------------------
Z   CY                   Z   CY
0   1   00100101  W      1   1   00000000  W
```

przy czym: (1) — przeniesienie
Reguły ustawienia wskaźników sygnalizacyjnych dla operacji odejmowania są następujące:

CY = 1 wtedy i tylko wtedy, gdy wystąpi pożyczka spoza tablicy A. W przeciwnym przypadku CY = 0.

Z = 1 wtedy i tylko wtedy, gdy wynik operacji odejmowania jest równy zeru (W = 0). W przeciwnym przypadku Z = 0.

Przykłady: W = A − B

	(1)	01110011	A			01110010	A
	−	10101001	B		−	00001101	B
Z	CY			Z	CY		
0	1	11001010	W	0	0	01100101	W

przy czym: (1) — pożyczka

Zauważmy, że dla operacji odejmowania wskaźnik CY = 1 sygnalizuje A < B, gdyż tylko wtedy wystąpi pożyczka, zaś wskaźnik CY = 0 sygnalizuje zależność A ≥ B, gdyż tylko wtedy pożyczka nie wystąpi.

Mikroprocesorowa realizacja operacji logicznych

Poza omówionymi wyżej operacjami arytmetycznymi wszystkie mikroprocesory potrafią wykonywać operacje logiczne na liczbach binarnych. Zdefiniujmy podstawowe operacje logiczne dla cyfr binarnych. Oznaczmy:

∧ — iloczyn logiczny (ang. AND — pol. „i")
∨ — suma logiczna (ang. OR — pol. „lub")
⊕ — suma „modulo 2" (ang. Exclusive OR — pol. „albo")

ILOCZYN LOGICZNY dwóch cyfr binarnych równy jest 1 wtedy i tylko wtedy, gdy obie cyfry są równe 1.

0 ∧ 0 = 0
0 ∧ 1 = 0
1 ∧ 0 = 0
1 ∧ 1 = 1

$$\equiv \begin{array}{cccc} 0 & 0 & 1 & 1 \\ \wedge\ \underline{0} & \wedge\ \underline{1} & \wedge\ \underline{0} & \wedge\ \underline{1} \\ 0 & 0 & 0 & 1 \end{array}$$

SUMA LOGICZNA dwóch cyfr binarnych równa jest 1 wtedy i tylko wtedy, gdy którakolwiek z cyfr równa jest 1.

0 ∨ 0 = 0
0 ∨ 1 = 1
1 ∨ 0 = 1
1 ∨ 1 = 1

$$\equiv \begin{array}{cccc} 0 & 0 & 1 & 1 \\ \vee\ \underline{0} & \vee\ \underline{1} & \vee\ \underline{0} & \vee\ \underline{1} \\ 0 & 1 & 1 & 1 \end{array}$$

SUMA „MODULO 2" dwóch cyfr binarnych równa jest 1 wtedy i tylko wtedy, gdy pierwsza ALBO druga cyfra równa jest 1.

0 ⊕ 0 = 0
0 ⊕ 1 = 1
1 ⊕ 0 = 1
1 ⊕ 1 = 0

$$\equiv \begin{array}{cccc} 0 & 0 & 1 & 1 \\ \oplus\ \underline{0} & \oplus\ \underline{1} & \oplus\ \underline{0} & \oplus\ \underline{1} \\ 0 & 1 & 1 & 0 \end{array}$$

Wykonanie operacji logicznej na dwóch liczbach binarnych polega na wykonaniu operacji logicznych na wszystkich cyfrach binarnych o tych samych wagach.

Większość osób operacje logiczne wykonuje kolumnami poczynając od cyfry najmniej znaczącej.

$$\begin{array}{r} 1010 \\ \wedge\ \underline{1011} \\ 1010 \end{array} \qquad \begin{array}{r} 1010 \\ \vee\ \underline{1011} \\ 1011 \end{array} \qquad \begin{array}{r} 1010 \\ \oplus\ \underline{1011} \\ 0001 \end{array}$$

Z ostatniego przykładu łatwo wydedukować, dlaczego operacja ⊕ nazywa się sumą modulo 2. Widać, że wykonujemy ją analogicznie, jak najzwyklejsze dodawanie dwóch liczb binarnych z tym, że nie uwzględniamy przeniesień.

Powróćmy teraz do naszych tablic A, B, W, CY, Z.

Reguły ustawiania wskaźników sygnalizacyjnych dla operacji logicznych są następujące:

CY = 0 ZAWSZE
Operacje logiczne zerują wskaźnik przeniesienia CY

Z = 1 wtedy i tylko wtedy, gdy wynik operacji jest równy zeru (W = 0). W przeciwnym przypadku Z = 0.

Przykłady:

		10101001	A			11101000	A
		01110011	B			00000111	B
Z	CY			Z	CY		
0	0	00100001	A ∧ B	1	0	00000000	A ∧ B
Z	CY			Z	CY		
0	0	11111011	A ∨ B	0	0	11101111	A ∨ B
Z	CY			Z	CY		
0	0	11011010	A ⊕ B	0	0	11101111	A ⊕ B

Proponujemy wykonać operacje arytmetyczne A + B, A − B oraz logiczne A ∧ B, A ∨ B, A ⊕ B dla wartości A, B podanych w poniższej tabelce. Uwzględnić CY i Z.

	1	2	3	4	5
A	55H	OFH	A9H	C8H	OBH
B	55H	FOH	B7H	38H	CDH

Co sygnalizuje wskaźnik Z = 1 dla operacji A − B oraz A ⊕ B?
Co sygnalizuje wskaźnik Z = 1 dla operacji A ∨ B?

Mnożenie i dzielenie liczb binarnych

W praktyce prostych sterowań operacje mnożenia i dzielenia są wykonywane bardzo rzadko. Analogia do liczb dziesiętnych jest oczywista. W tablicy 4 przedstawiono przykłady „papierkowego" mnożenia i dzielenia.

Jak widać, operację mnożenia liczb binarnych można zastąpić operacjami dodawania, zaś operację dzielenia liczb binarnych — operacjami odejmowania.

Zróbmy dalsze porównania.

$$\begin{array}{l} \underline{2{,}3(3)} \\ \underline{7:3} \\ -\ \underline{6} \\ 10 \\ \underline{9} \\ (3) \end{array} \qquad 2{,}3(3) = 2\cdot 10^{0} + 3\cdot 10^{-1} + 3\cdot 10^{-2} + 3\cdot 10^{-3}\ldots$$

gdzie: $10^{-1} = \frac{1}{10^{1}}$, $10^{-2} = \frac{1}{10^{2}}$....

(3) — w okresie

Tablica 4. **Przykłady mnożenia i dzielenia liczb binarnych**

Dziesiętnie	Binarnie	Sprawdzenie
7 × 5 35	111 × 101 111 + 111 100011	100011B = 32 + 2 + 1 = 35
5 35:7 − 35 00	101 100011:111 − 111 00111 − 111 000	101B = 4 + 1 = 5

Powyższe, to nic innego jak zapis ogólny liczby dzisiętnej rozciągnięty na część po przecinku. Zróbmy to samo co wyżej na liczbach binarnych (7 = 111B, 3 = 11B).

```
 10,01(01)
 ---------
 111 : 11
-  11
 -------
 00100
   - 11
   -----
    0100
    - 11
    -----
     (01) — w okresie
```

$$10{,}01(01) = 1\cdot 2^{1} + 0\cdot 2^{0} + 0\cdot 2^{-1} + 1\cdot 2^{-2} + 0\cdot 2^{-3} + 1\cdot 2^{-4} + \ldots\ldots$$

gdzie: $2^{-1} = \frac{1}{2^{1}}$, $2^{-2} = \frac{1}{2^{2}}$.....

Zamieńmy liczbę binarną 10,01(01) na liczbę dziesiętną sprawdzając, czy otrzymamy 2,3(3).

$$\begin{aligned}
&1\cdot 2^{1} = 2\\
&1\cdot 2^{-2} = 0{,}25\\
&1\cdot 2^{-4} = 0{,}0625\\
&1\cdot 2^{-6} = 0{,}015625\\
+\ &1\cdot 2^{-8} = 0{,}0039625\\
&\vdots\\
&\overline{\underline{2{,}3320875}}
\end{aligned}$$

Jeżeli ktoś nie wierzy, że liczba binarna 10,01(01) odpowiada liczbie dziesiętnej 2,3(3), może kontynuować powyższe dodawanie dążąc do... nieskończoności.

Doskonale wiemy, że aby pomnożyć liczbę dziesiętną przez 10^{1}, 10^{2}, 10^{3}... wystarczy przesunąć przecinek o 1, 2, 3... pozycje w prawo, zaś aby pomnożyć liczbę dziesiętną przez 10^{-1}, 10^{-2}, 10^{-3}... wystarczy przesunąć przecinek o 1, 2, 3... pozycje w lewo.

Można samemu sprawdzić, że mnożenie liczby binarnej przez $2^{1}, 2^{2}, 2^{3}$... odpowiada przesunięciu przecinka o 1, 2, 3... pozycje w prawo, zaś mnożenie liczby binarnej przez 2^{-1}, 2^{-2}, 2^{-3}... odpowiada przesunięciu przecinka o 1, 2, 3... pozycje w lewo.

Wykonanie poniższych działań arytmetycznych na liczbach binarnych umożliwi sprawdzenie waszych umiejętności. W nawiasach podano liczby dziesiętne odpowiadające liczbom binarnym. 110101 (53) × 1101 (13), 1010110001 (689) : 1101 (13)

□

mgr inż. Stanisław Gardynik

Podstawy techniki mikroprocesorowej (3)

Fizyczna interpretacja liczb binarnych

Najpopularniejszą fizyczną interpretacją pojedynczej cyfry binarnej jest sygnał w standardzie TTL (rys. 1), w którym: cyfrze binarnej „0" odpowiada napięcie w granicach 0 ÷ 0,4 V zaś cyfrze binarnej „1" odpowiada napięcie 2,4 ÷ 5 V.
W technice TTL do oznaczenia cyfry binarnej „0" używa się często litery L, zaś do oznaczenia cyfry binarnej „1" litery „H".
L (ang. LOW) — niski poziom TTL (0 ÷ 0,4 V)
H (ang. HIGH) — wysoki poziom TTL (2,4 ÷ 5 V)
Za pomocą tranzystorów i rezystorów sygnały TTL realizowane są bardzo prosto. Zobaczmy to na przykładzie najprostszego elementu logicznego jakim jest inwerter (rys. 2).
Inwerter realizuje najprostszą funkcję logiczną jaką jest negacja sygnału wejściowego:

$Y = \overline{A}$ — symbol negacji

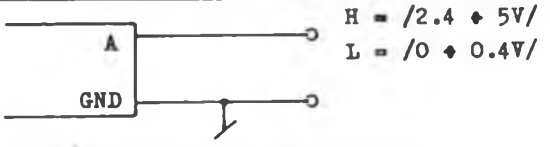
przy czym:
A — cyfra binarna, tzn. może przyjmować wartości wyłącznie „0" albo „1".
Definicja negacji to nic innego jak tabela z rys. 2d.
Można łatwo zbudować najprostszy układ inwertera i sprawdzić, że rzeczywiście:

a) Jeśli do wejścia A doprowadzimy niski poziom logiczny (0 ÷ 0,4 V), to na wyjściu Y otrzymamy wysoki poziom logiczny (2,4 ÷ 5 V) (tranzystor będzie w stanie odcięcia — $V_Y = 5$ V) $Y = \overline{L} = H \equiv Y = \overline{0} = 1$
b) Jeśli do wejścia A doprowadzimy wysoki poziom logiczny (2,4 ÷ 5 V) to na wyjściu Y otrzymamy niski poziom logiczny (0 ÷ 0,4 V) (tranzystor będzie w stanie nasycenia — $V_Y = U_{CESAT} \cong 0{,}2$ V)
$Y = \overline{H} = L \equiv Y = \overline{1} = 0$

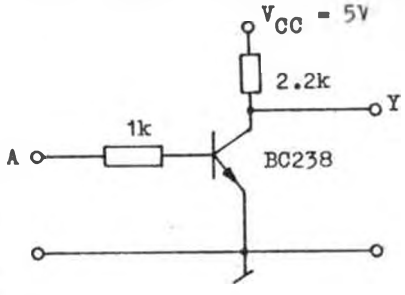
Wszystkie nowoczesne układy mikroprocesorowe (tj. mikroprocesory, pamięci, układy wejścia/wyjścia itp.) mają wyprowadzone sygnały całkowicie kompatybilne z sygnałami TTL.
Zapamiętaj! $L \equiv 0$ $H \equiv 1$

Znamy już fizyczną interpretację cyfry binarnej. Wiemy, że pojedynczy przewód elektryczny, na którym mogą wystąpić tylko dwa stany można traktować jako cyfrę binarną.
Liczba binarna, to uporządkowany ciąg cyfr binarnych. Fizyczną interpretacją liczby binarnej będzie zatem uporządkowany zbiór przewodów — cyfr binarnych (rys. 3).
Na każdym z przewodów A0, A1, A2, A3 może wystąpić napięcie wyłącznie H = (2,4 ÷ 5 V) albo L = (0 ÷ 0,4 V) względem masy. W technice cyfrowej prawie nigdy nie uwidocznia się przewodu masy, gdyż oczywiste jest, że poziomy napięć na liniach sygnałowych są mierzone względem masy.

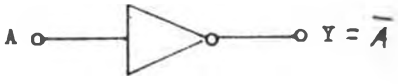
Przewody sygnałowe łączące dwa układy cyfrowe należy zawsze traktować jako cyfry lub liczby binarne.



Rys. 1. Fizyczna interpretacja cyfry binarnej



a



b

c

V_A	V_Y
0V	5V
5V	0V

d

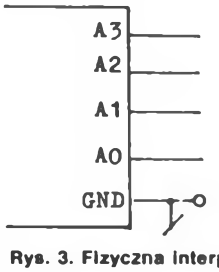
A	$Y=\overline{A}$
0	1
1	0

Rys. 2. Realizacja inwertera
a — układ elektryczny inwertera, b — symbol graficzny inwertera, c — napięcia na zaciskach układu, d — tablica prawdy

Jeśli między dwoma układami cyfrowymi są przesyłane liczby binarne, to przewody przez które liczby te są przesyłane muszą być połączone w ściśle określonym porządku — w sposób przedstawiony na rys. 4.

Układ Z80A CPU jest nadajnikiem liczby binarnej, zaś układ 8255 odbiornikiem liczby binarnej. Osiem linii przesyłowych D7 ÷ D0 reprezentuje liczbę binarną.
$D_7D_6D_5D_4D_3D_2D_1D_0 = D_7 \cdot 2^7 + D_6 \cdot 2^6 + \ldots + D_1 \cdot 2^1 + D_0 \cdot 2^0$

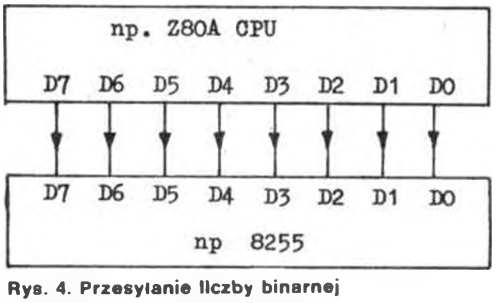
Zbiór liniii służących do przesyłania liczb binarnych nazywamy często MAGISTRALĄ (ang. BUS), zaś pojedyncze linie reprezentujące cyfry binarne są nazywane SYGNAŁAMI CYFROWYMI.



Reprezentowana liczba binarna

$A_3A_2A_1A_0 = A_3 \cdot 2^3 + A_2 \cdot 2^2 + A_1 \cdot 2^1 + A_0 \cdot 2^0$

Rys. 3. Fizyczna interpretacja liczby binarnej



Rys. 4. Przesyłanie liczby binarnej

Przyjrzyjmy się bliżej fizycznej realizacji prostego nadajnika i odbiornika cyfry binarnej z rys. 5.
Jak widać, na linii przesyłowej sygnał cyfrowy jest zanegowany ale po minięciu odbiornika jest identyczny z tym, który nadaje nadajnik. Przy okazji zapoznaliśmy się z fizyczną interpretacją jednego z podstawowych twierdzeń algebry dwuelementowej, tzw. algebry Boole'a.

$$\overline{\overline{A}} = A$$

przy czym: A — cyfra binarna
Twierdzenia algebry Boole'a dowodzi się niezwykle prosto. Należy po prostu udowodnić prawdziwość wszystkich możliwych przypadków co przedstawiono w tablicy 1. Powyższą tablicę wypełniono korzystając z definicji negacji. Widać z niej, że: $A = \overline{\overline{A}}$, co należało udowodnić.

Tablica 1. **Dowód twierdzenia algebry Boole'a o negacji**

A	$Y_1 = \overline{A}$	$Y_2 = \overline{Y_1} = \overline{\overline{A}}$
0	1	0
1	0	1

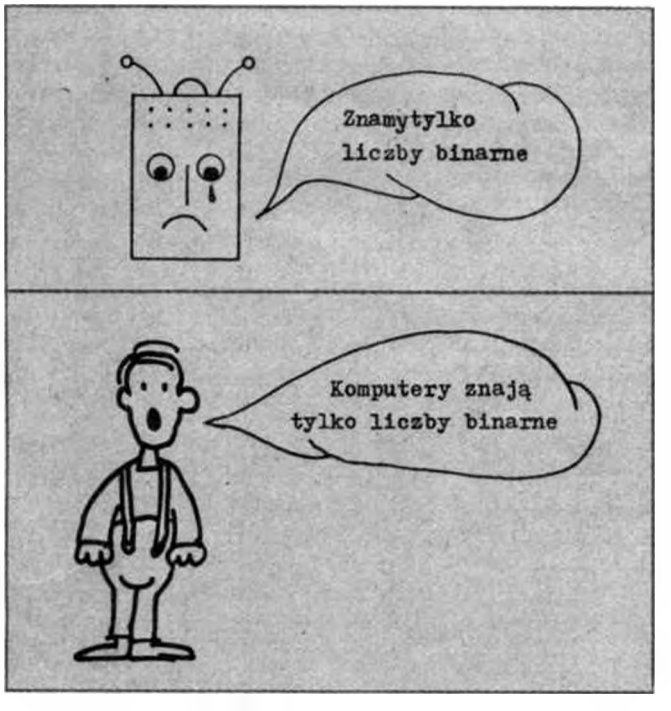
$A = \{0, 1\}$ — cyfra binarna

Dwuelementową algebrą Boole'a posługujemy się na codzień, chociaż nie zawsze zdajemy sobie z tego sprawę.
Porównajmy dwa zadania:
1. Nie jestem nieuczciwy
2. Jestem uczciwy
Widać, że są one równoważne. W pierwszym z nich zanegowaliśmy dwa razy wyraz uczciwy. W drugim pominęliśmy te dwie negacje i otrzymaliśmy to samo.

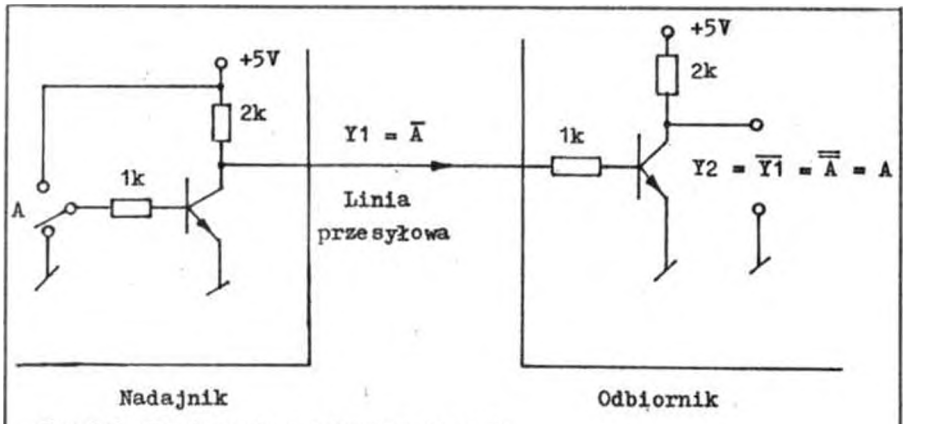
$\overline{\overline{\text{UCZCIWY}}} \equiv$ uczciwy

Komunikacja ze światem zewnętrznym

Komputery twierdzą:
Z naszego punktu widzenia nieprawdą jest, że



Wszyscy wiemy, że potrafią one „czytać", „rysować" itp. Przyjrzyjmy się bliżej w jaki sposób komputery „czytają" i „piszą".
Każdy chyba wie, że z komputerem człowiek najczęściej komunikuje się za pomocą klawiatury. Znacznie mniej osób wie, że z klawiatury do komputera są transmitowane znaki alfanumeryczne zupełnie niepodobne do tych, które człowiek widzi. Są to po prostu LICZBY BINARNE.
Każdy znak alfanumeryczny widziany przez człowieka na klawiszach klawiatury ma swój odpowiednik w liczbie binarnej. Obecnie na całym świecie stosuje się jeden, standardowy kod, według którego koduje się znaki alfanumeryczne. Nosi on nazwę kodu ASCII i przedstawiony jest w tablicy 2.
Spróbujmy samodzielnie odczytać binarne kody kilku liter, np.: A, G, Z ... Widoczne w tablicy kilkuliterowe nazwy, to

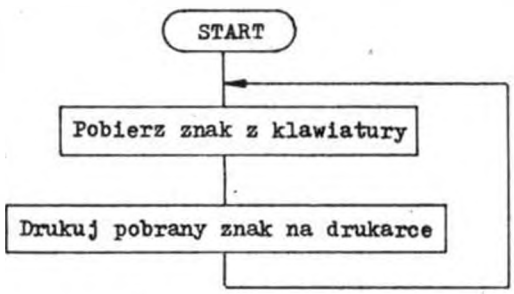


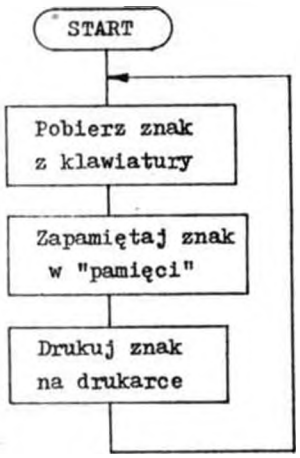
Rys. 5. Realizacja prostego nadajnika i odbiornika

Tablica 2. **Kod ASCII**

HEX LSD	MSC BITS	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	0000	NUL	DLE	SPACE	0	@	P	–	p
1	0001	SOH	DC1	1	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	S
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	--
D	1101	CR	GS	–	=	M	]	m	}
E	1110	SO	RS	.	>	N	Λ	n	~
F	1111	SI	US	/	?	O	←	o	DEL

symbole pewnych operacji, z których najważniejsze, to podstawowe rozkazy dla drukarki (maszyny do pisania).
SPACE — zrób odstęp (spację)
CR — zacznij pisać od początku linii
LF — zacznij pisać od nowej linii (wysunięcie papieru o 1 linie)
Nie trzeba chyba przekonywać, że taki a nie inny sposób ustawienia znaków w powyższej tabeli jest sprawą umowną, że równie dobrze mogłyby one być poustawiane inaczej.
Określmy algorytm programu „maszyny do pisania", który pobiera znaki z klawiatury alfanumerycznej i drukuje je na drukarce. Przedstawiono go na rys. 6.
To wszystko. Mamy program, który chcielibyśmy mieć. Zakodowanie powyższego programu w postaci rozkazów dla





Rys. 6. Algorytm programu pisania na maszynie

Rys. 7. Algorytm programu pisania na maszynie z zapamiętaniem tekstu

komputera to zwykła formalność — oczywiście po zapoznaniu się z listą rozkazów danego komputera. Widać, że komputer wykonuje „nieskończoną" pętlę programową, w której cały czas pobiera znaki z klawiatury i drukuje je na drukarce. Załóżmy teraz, że chcemy mieć bardziej wyrafinowaną maszynę do pisania, która potrafiłaby powielać raz napisany tekst dowolną liczbę razy. Jest chyba oczywiste, że aby wykonać dowolną liczbę kopii komputer musi zapamiętać wszystkie klawisze, które zostały naciśnięte, zgodnie z algorytmem z rys. 7.

Wyobraźmy sobie, że do komputera wykonującego powyższy program podszedł człowiek i zaczął coś pisać. Pozwólmy mu napisać kilka znaków, po czym zobaczmy co jest w pamięci komputera. Przedstawiono to w tablicy 3.

Tablica 3. **Obraz pamięci komputera zawierającej wpisany z klawiatury tekst**

Klawisze wciśnięte przez człowieka	Pamięć komputera BIN (kod ASCII)	Pamięć komputera HEX (kod ASCII)
C	01000011	43
P	01010000	50
L	01001100	4C
CR	00001101	OD
LF	00001010	OA
L	01001100	4C
D	01000100	44
SPAC	00100000	20
A	01000001	41
,	00101100	2C
B	01000010	42

Kolejność wciskania klawiszy

CPL LD A,B

Aktualny stan drukarki

Widać, że znajdujemy się w chwili, w której komputer „dopiero co" przyjął i wydrukował literę „B", że człowiek pisze coś w języku, którego nie rozumiemy. Język jest w tym przypadku nieistotny. Najważniejsze, abyśmy czuli związek między naciskaniem klawiszy, a tym co się dzieje w pamięci komputera i na drukarce.

Zauważmy że:

1. Z klawiatury do komputera „płyną" znaki alfanumeryczne zakodowane w kodzie ASCII, czyli ... liczby binarne.
2. W pamięci komputera są przechowywane kody ASCII (liczby binarne) wciśniętych znaków alfanumerycznych.
3. Komputer drukuje na drukarce ciąg znaków alfanumerycznych (tekst) przesyłając do niej odpowiednie kody ASCII, czyli ... liczby binarne. Drukarka zamienia te kody na znaki alfanumeryczne czytelne dla człowieka.

Program, który umożliwia wprowadzenie dowolnego tekstu (ciągu znaków alfanumerycznych) do pamięci komputera nazywa się programem EDYTORA. Dobre programy EDYTORA umożliwiają nie tylko wprowadzenie tekstu, ale również łatwą jego modyfikację.

Proponujemy zastanowić się jak będzie wyglądać tekst z tablicy 3 na drukarce, jeśli z pamięci komputera usuniemy liczbę binarną będącą kodem symbolu CR?

Innym istotnym z punktu widzenia komunikacji ze światem zewnętrznym jest kod BCD. Kod BCD umożliwia niezwykle proste kodowanie liczb dziesiętnych zgodnie z tablicą 4.

Tablica 4. **Kod BCD**

Zapis binarny	0000	0001	0010	0011	0100
Liczba BCD	0	1	2	3	4

Zapis binarny	0101	0110	0111	1000	1001
Liczba BCD	5	6	7	8	9

Zakodujmy kilka liczb dziesiętnych w kodzie BCD:

$$74 = \underset{0111}{7}\ \underset{0100}{4} \qquad 69 = \underset{0110}{6}\ \underset{1001}{9}$$

$$1985 = \underset{0001}{1}\ \underset{1001}{9}\ \underset{1000}{8}\ \underset{0101}{5} \qquad 320 = \underset{0011}{3}\ \underset{0010}{2}\ \underset{0000}{0}$$

Liczby dziesiętne zakodowane w kodzie BCD i przechowywane w pamięci komputera można w bardzo prosty sposób drukować na drukarce. Zauważmy, że aby wydrukować dowolną liczbę, wystarczy pobierać kolejne cyfry zakodowane w kodzie BCD, dodawać do nich liczbę 30H (zamiana na kod ASCII) i wysyłać na drukarkę.

Np.
```
  00001001 — cyfra 9
+ 00110000 — liczba 30H
  --------
  00111001 — kod ASCII cyfry 9
```

Stosowanie kodu BCD zamiast binarnego umożliwia wykonywanie prostych operacji arytmetycznych jak gdyby bezpośrednio na liczbach dziesiętnych. Nowoczesne mikroprocesory umożliwiają przeprowadzenie obliczeń bezpośrednio w kodzie BCD, jak na przykład:

Dodawanie w kodzie BCD	To samo dziesiętnie	Dodawanie w kodzie BCD	To samo dziesiętnie
00110101 A	35	00010010 A	12
+ 00110001 B	+31	+ 01011001 B	+59
a. 01100110 W	66	b. 01110001 W	71

□

mgr inż. Stanisław Gardynik

Podstawy techniki mikroprocesorowej (4)

Budowa mikroprocesora

Najogólniej można stwierdzić, że mikroprocesor jest „czarną skrzynką" znakomicie operującą liczbami binarnymi. To „operowanie" polega na przesyłaniu liczb binarnych oraz wykonywaniu na nich prostych operacji arytmetycznych i logicznych. W swoim wnętrzu potrafi on zapamiętać zaledwie 16 liczb ośmiobitowych, 4 szesnastobitowe oraz kilka cyfr binarnych pełniących funkcję wskaźników dla potrzeb operacji arytmetycznych i logicznych.
W czym więc jego siła? Co ciekawego można zrobić z tak niewielką ilością liczb?
Odpowiedź jest jedna: NIC. „Goły" mikroprocesor wart jest mniej więcej tyle, ile silnik bez samochodu. Cała tajemnica tkwi w tym, że potrafi on sterować bardzo dużą pamięcią zewnętrzną o pojemności do 65536 ośmiobitowych liczb binarnych.

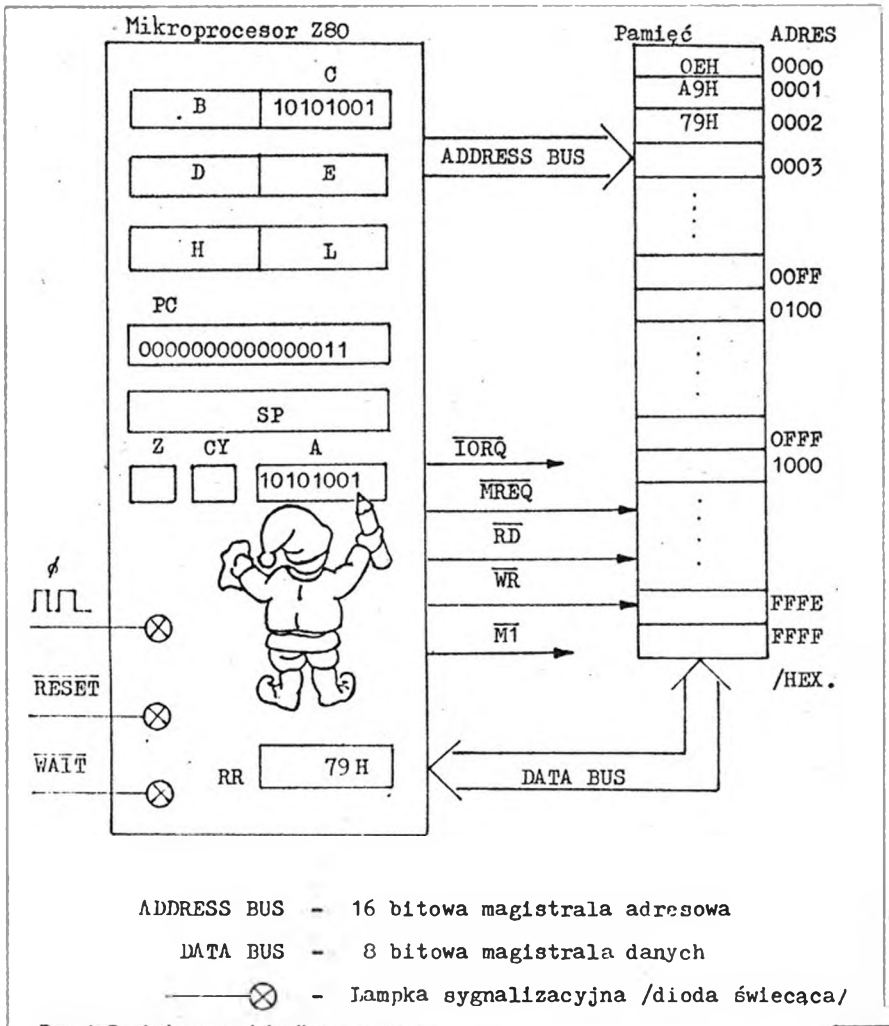
Co jest w środku mikroprocesora?

W środku mikroprocesora znajduje się kilka tysięcy tranzystorów realizujących ... Myślę, że nikt z czytelników nie ma ochoty na oglądanie dokładnego schematu ideowego mikroprocesora. Spójrzmy więc nieco inaczej na mikroprocesor. Przecież z punktu widzenia zarówno projektanta systemów mikroprocesorowych jak i programisty, istotny jest fakt, że mikroprocesor potrafi wykonywać operacje na liczbach binarnych. Wyobraźmy sobie schemat ideowy mikroprocesora Z80 wraz z niezbędną pamięcią zewnętrzną w postaci przedstawionej na rys. 1.
Zapewne wielu czytelników na widok powyższego schematu przyżyje szok. Co tam robi krasnoludek?
Dla wyjaśnienia przeanalizujmy dwa stwierdzenia:
— Mikroprocesor wykonuje operację dodawania
— Krasnoludek wykonuje operację dodawania
Pierwsze zdanie rodzi w człowieku niepotrzebną ciekawość — jak on to robi?
Drugie zdanie nie stwarza takiej sytuacji. Po prostu... krasnoludek to potrrafi!
Drugim argumentem za krasnoludkiem jest uatrakcyjnienie nudnych czasami opisów książkowych. Niech krasnoludek pracujący w mikroprocesorze ma na imię MIK. Zobaczmy jak przy jego pomocy można wyjaśnić pracę mikroprocesora. Nasz krasnoludek ma do dyspozycji kilkanaście szkolnych tablic, kredę i ścierkę.
Na tablicach o nazwach A, B, C, D, E, H, L, RR może zapisywać wyłącznie 8 bitowe liczby binarne, na PC, SP szesnastobitowe, zaś na małych tabliczkach CY i Z zmieści mu się tylko 1 cyfra binarna. Nadmiernie dociekliwi mogą sobie dorysować na wszystkich liniach wejściowych „lampki" zaś na wyjściowych przełączniki (rys. 2).
MIK nie ma więc żadnych problemów z odczytywaniem i wysyłaniem sygnałów cyfrowych.
W dalszej części opisu zamiast słowa „tablica" będziemy często używali słowa „rejestr" zaś zamiast „krasnoludek" słowa „mikroprocesor" lub MIK.

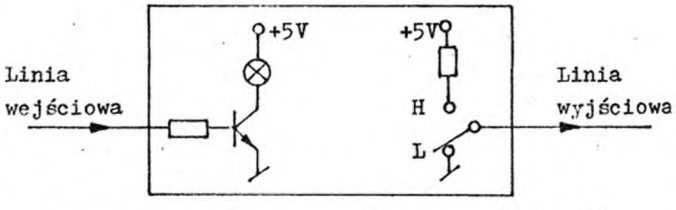
Na rys. 1 przedstawiono tylko najważniejsze rejestry wewnętrzne mikroprocesora Z80. Celowo pominięto rejestry rzadziej używane, aby nie zaciemniać obrazu mikroprocesora początkującemu mikroelektronikowi. Dobre zrozumienie zasady działania powyższego, „zubożonego" Z80 gwarantuje zrozumienie zasady wszystkich mikroprocesorów. Należy zdawać sobie sprawę z faktu, że MIK nie jest niewolnikiem, że bardzo lubi i chce pracować.

O szybkości pracy mikroprocesora decyduje zewnętrzny zegar, sygnał o postaci przedstawionej na rys. 3. Człowiek może regulować ten zegar wpływając na szybkość pracy krasnoludka. Możliwości mikroprocesora w operowaniu liczbami są dużo większe niż możliwości człowieka. Krasnoludek z Z80 potrafi wykonać około 600 000 prostych operacji na liczbach binarnych w ciągu 1 sek.

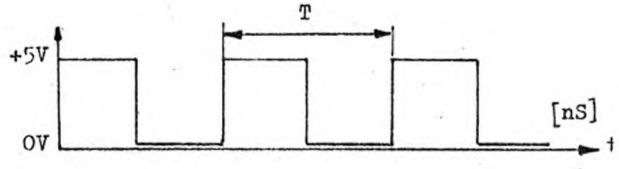
Jeden takt zegara dla mikroprocesora Z80 CPU może trwać od 400 do 4000 ns. Produkowane są również szybsze wersje co pokazuje tablica 1. Wersje mikroprocesora Z80 różnią się wyłącznie szybkością wykonywania operacji na liczbach binarnych. Na tablicach znajdujących się w środku mikroprocesora krasnoludek może wykonywać operacje arytmetyczne i logiczne oraz operacje przesłania między rejestrami.



Rys. 1. Poglądowy model mikroprocesora

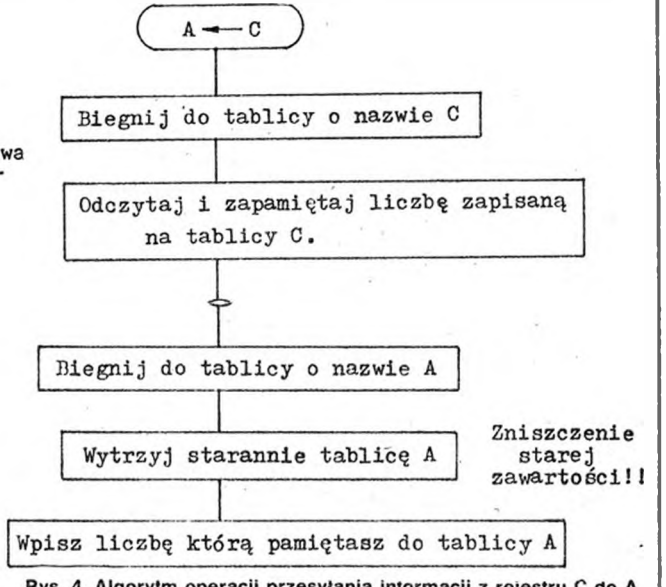


Rys. 2. Sposób odbierania sygnałów wejściowych i oddziaływania na sygnały wyjściowe przez MIK-a



T - okres zegara zwany często taktem

Rys. 3. Postać sygnału zegara procesora



Rys. 4. Algorytm operacji przesyłania informacji z rejestru C do A.

Tablica 1. **Parametry różnych wersji procesora Z80**

Mikro-procesor	f_{max} [MHz]	f_{min} [MHz]	$T_{min} = \frac{1}{f_{max}}$	Liczba prostych operacji na sekundę
Z80 CPU	2,5	0,25	400	625 000
Z80A CPU	4	0,25	250	1 000 000
Z80B CPU	6	0,25	166	1 562 000
Z80C CPU	8	0,25	125	2 000 000

Zobaczmy na przykładzie operacji przesłania w jaki sposób MIK wykonuje działania.

Algorytm przesłania zawartości rejestru (tablicy) C do rejestru (tablicy) A przedstawiono na rys. 4.

Operacja przesłania jest zapisywana symbolicznie następująco:

$$A \leftarrow C \quad \text{lub} \quad A := C$$

Rejestr z którego jest pobierana informacja nosi nazwę rejestru źródłowego (ang. source register), zaś rejestr do

którego informacja jest wpisywana nosi nazwę rejestru przeznaczenia (ang. destination register).
W powyższym przykładzie rejestrem źródłowym jest rejestr C zaś rejestrem przeznaczenia rejestr A.
Należy pamiętać że:

1. Operacja przesłania nie zmienia zawartości rejestru źródłowego.
2. Operacja przesłania niszczy zawartość rejestru przeznaczenia.

Ośmiobitowe rejestry B, C, D, E, H, L są zwane rejestrami ogólnego przeznaczenia. Rejestr o nazwie A pełni szczególną rolę przy wykonywaniu operacji arytmetycznych i logicznych i nazywany jest akumulatorem.
W mikroprocesorze Z80 możliwe są następujące operacje przesłania między rejestrami 8-bitowymi.

$$r \leftarrow s \quad \text{gdzie: } r, s \in \{A, B, C, D, E, H, L\}$$

Firma Zilog producent Z80 CPU) stosuje następujący zapis słowny dla powyższej operacji przesłania

LD r, s (ŁADUJ ang. LOAD)

W literaturze zapis słowny rozkazu nazywa się mnemonikiem.
Niektóre rozkazy mikroprocesora Z80 traktują dwa ośmiobitowe rejestry ogólnego przeznaczenia jako jeden rejestr 16-bitowy. Następujące pary rejestrów 8-bitowych mogą być traktowane jako jeden rejestr 16-bitowy:

BC, DE, HL.

Nie przypadkowo więc na schemacie ideowym z rys. 1 rejestry ogólnego przeznaczenia narysowano w taki, a nie inny sposób. Dla mikroprocesora istnieją tylko dwa rozkazy dotyczące przesłań między rejestrami 16-bitowymi.

$PC \leftarrow HL$ JP (HL)
$SP \leftarrow HL$ LD SP, HL

Należy zauważyć, że przesłania między rejestrami szesnastobitowymi BC, DE, HL można łatwo dokonać przy pomocy dwóch prostych przesłań 8-bitowych jak na przykład:

$$BC \leftarrow DE \equiv \begin{matrix} C \leftarrow E \\ B \leftarrow D \end{matrix}$$

Zdecydowana większość najczęściej używanych rozkazów mikroprocesora Z80 jest identyczna jak w procesorach 8080, 8085. Niestety, większość tych rozkazów ma inne zapisy słowne. Kilka przykładów podano w tablicy 2.

Tablica 2. **Przykłady zapisów symbolicznych i słownych różnych rozkazów.**

Zapis symboliczny	Zapis słowny	
	Zilog	Intel
$A \leftarrow C$	LD A, C	MOV A, C
$PC \leftarrow HL$	JP (HL)	PCHL
$A \leftarrow A+1$	INC A	INR A
$A \leftarrow \bar{A}$	CPL	CMA
$A \leftarrow A \wedge C$	AND C	ANA C

Jest to jeden z przykładów pt. „W jaki sposób ludzie utrudniają sobie życie?" To utrudnienie jest wynikiem szalejącej konkurencji. Mikroprocesor 8080 był bowiem pierwszym przyzwoitym mikroprocesorem 8-bitowym opracowanym przez firmę Intel w 1974 r. Główni autorzy, kilku zdolnych ludzi, po sukcesie.. wymówili pracę Intelowi, założyli własną firmę (Zilog) i wylansowali udoskonaloną i rozszerzoną wersję 8080 — Z80. Mikroprocesor ten po dziś dzień wydaje się być królem mikroprocesorów w swojej klasie.
W podobny sposób powstał deszcz firm zajmujących się mikroelektroniką i.. szalejącą konkurencją. Firmy będące pod jej wpływem zmuszone są do niezwykle intensywnego udoskonalania swoich wyrobów, gdyż w przeciwnym razie grozi im... bankructwo. Szalejąca konkurencja jest jednym z głównych motorów napędowych rozwoju mikroelektroniki.

□

mgr inż. Stanisław Gardynik

Podstawy techniki mikroprocesorowej (5)

Współpraca mikroprocesora z pamięcią

Rozważmy, w jaki sposób mikroprocesor komunikuje się z pamięcią zewnętrzną. Posłużymy się w tym celu modelem połączenia procesora z pamięcią przedstawionym na rys. 1 i załóżmy, że procesor może wykonać tylko dwie elementarne operacje:

— odczyt pamięci — MR (ang. Memory Read)

— zapis pamięci — MW (ang. Memory Write).

Mikroprocesor może za pośrednictwem magistrali adresowej zaadresować dowolną komórkę pamięci ustawiając na niej 16-bitową liczbę binarną — adres jednej z 65536 (2^{16}) komórek pamięci. W każdej komórce pamięci zapisana jest jedna, 8-bitowa liczba binarna.

Realizując operację odczytu procesor pobiera 8-bitową liczbę binarną z zaadresowanej komórki za pośrednictwem magistrali danych. Przy operacji zapisu ustawia na magistrali danych 8-bitową liczbę binarną, która zostaje zapisana do zaadresowanej komórki pamięci niszcząc jej starą zawartość.

Kierunek przepływu informacji na magistrali danych zależy od stanu sygnałów sterujących. Wszystkie możliwe stany, jakie może wygenerować mikroprocesor, przedstawiono na rys. 2.

Wysoki poziom logiczny sygnału $\overline{MREQ}$ odcina pamięć od magistrali danych D0÷D7. Przewody magistrali dochodzą do pamięci, lecz w środku znajdują się „elektroniczne wyłączniki", które odłączają magistralę w przypadku $\overline{MREQ}$ = H. Przedstawiono to obrazowo na rys. 3.

Odłączenie pamięci jest konieczne np. w sytuacji np. w sytuacji, gdy mikroprocesor komunikuje się z układami wejścia/wyjścia. W literaturze fachowej stan odcięcia magistrali danych jest określany również jako „stan trzeci" lub „stan wysokiej impedancji" i oznaczany literą Z.

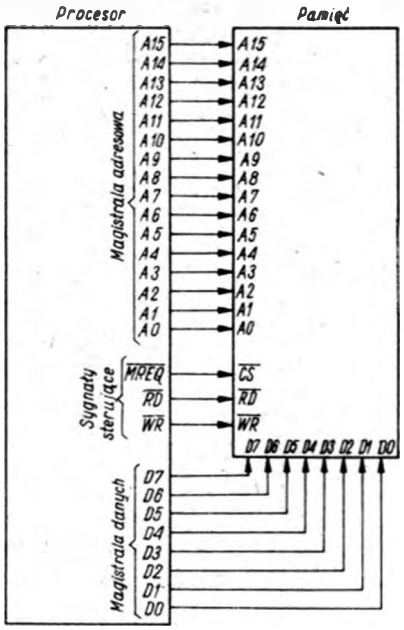
Cykl odczytu pamięci

Zajrzyjmy do środka mikroprocesora i zobaczmy jak nasz krasnoludek MIK realizuje operację odczytu. Algorytm postępowania przedstawiono w pamięci schematu blokowego na rys. 4. Założono, że sygnały sterujące w chwili wejścia do poniższego algorytmu są nieaktywne, tzn. $\overline{MREQ} = \overline{RD} = \overline{WR}$ = H. Trzeba dodatkowo przypomnieć, że MIK wykonuje wszystkie czynności w takt zewnętrznego zegara Φ. Operację odczytu można o wiele dokładniej przedstawić za pomocą wykresu czasowego, przedstawionego na rys. 5. W mikrokmputerze CA80 zegar mikroprocesora ma częstotliwość

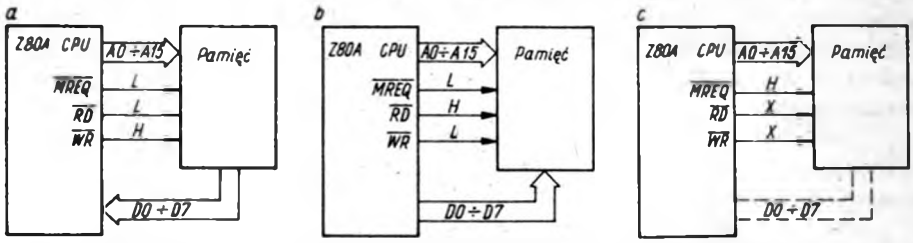
$$\Phi = 4\ \text{MHz}$$

Częstotliwości tej odpowiada okres

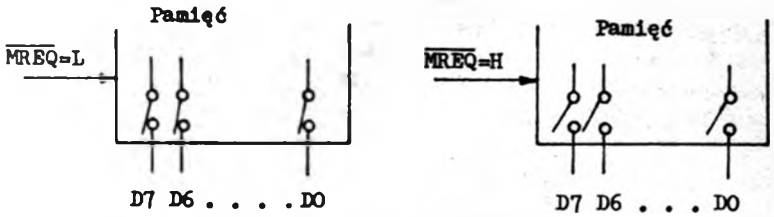
$$T = \frac{1}{\Phi} = \frac{1}{4\ \text{MHz}} = 250\ \text{ns}$$



Rys. 1 Sposób połączenia procesora z pamięcią
$\overline{MREQ}$ — sygnał zadania dostępu do pamięci (ang. memory request) $\overline{RD}$
$\overline{RD}$ — sygnał sterujący odczytem (ang. read) $\overline{WR}$ — sygnał sterujący zapisem (ang. write)

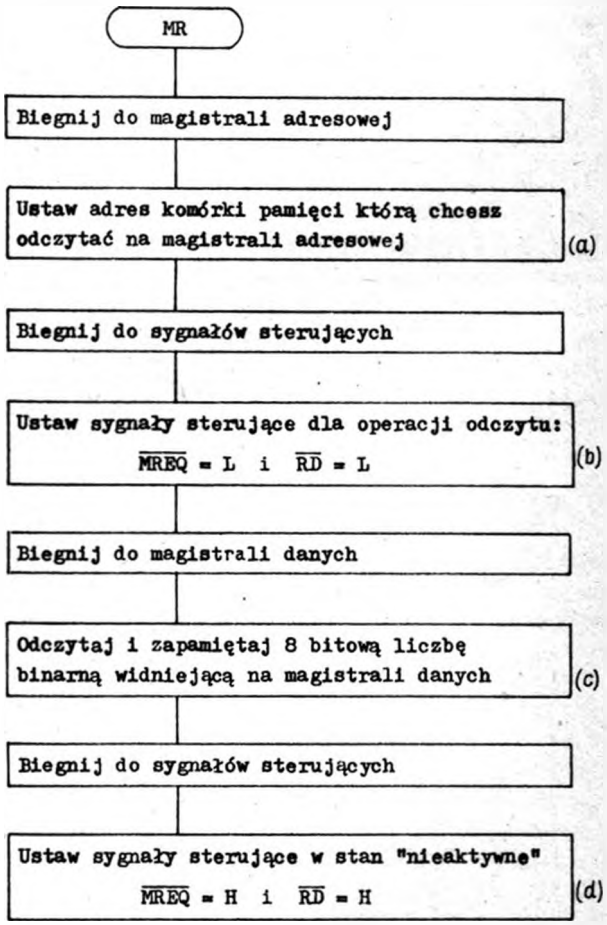


Rys. 2. Tryby współpracy procesora z pamięcią a — odczyt pamięci, b — zapis pamięci, c — odcięcie pamięci od procesora (H — stan wysoki, L — stan niski, X — stan dowolny)



Rys. 3. Stan odcięcia pamięci od procesora

Na wykresie czasowym widać, że mikroprocesor rozpoczyna operację odczytu od ustawienia 16-bitowej liczby binarnej na magistrali adresowej. Liczba ta jest adresem komórki pamięci, którą MIK chce odczytać. Dzieje się to na początku taktu zegarowego T1, ok. 40 ns po narastającym zboczu zegara w takcie T1. Wraz z opadającym zboczem zegara w takcie T1 uaktywniane są sygnały sterujące odczytem w stan aktywny:
$\overline{MREQ} = \overline{RD} = L$
Aktywne sygnały odczytu inicjują w pamięci zewnętrznej operację poszukiwania 8-bitowej liczby binarnej, której adres widnieje na magistrali adresowej. Czas konieczny do odszukania tej liczby i wystawienia jej na magistralę danych nazywany jest czasem dostępu do pamięci i oznaczany t_{ACC} (ang. access time).
W czasie poszukiwania 8-bitowej liczby binarnej informacja na magistrali danych może być niestabilna, tzn. mogą pojawić się na niej fałszywe liczby binarne, co zostało zaznaczone linią przerywaną. Mikroprocesor odczytuje 8-bitową liczbę binarną wraz z opadającym zboczem zegara w takcie T3. Operacja odczytu następuje po upływie ok. 500 ns od chwili uaktywnienia sygnałów sterujących odczytem. Aby odczyt był prawidłowy, musi być zatem spełniony warunek $t_{ACC} \leqslant 500$ ns. Chwilę po odczytaniu informacji mikroprocesor ustawia sygnały sterujące odczytem w stan nieaktywny: $\overline{MREQ} = \overline{RD} = H$
Zauważmy, że sygnał sterujący zapisem $\overline{WR}$ jest nieaktywny przez cały czas trwania operacji odczytu.
Nikt chyba nie ma wątpliwości, że z wykresu czasowego można odczytać o wiele więcej informacji, niż uzyskać na podstawie najlepszego nawet opisu słownego. Wykresy czasowe są międzynarodowym językiem, który musi być zrozumiały dla każdego mikroelektronika. Są one najlepszym sposobem opisu działania cyfrowych układów synchronicznych tj. układów, w których wszystkie istotne operacje są wykonywane w takt zewnętrznego zegara.
Należy zauważyć, że wszystkie podawane w katalogach parametry czasu są określane przeważnie względem tego właśnie sygnału.
Wyjaśnimy jak należy interpretować dane katalogowe na przykładzie przedstawionym w tablicy 1.

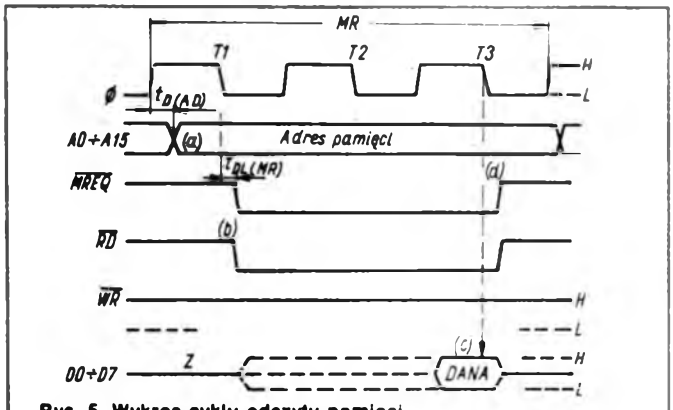


Rys. 4. Algorytm postępowania MIKa przy odczycie pamięci

Tablica 1. **Przykładowe dane katalogowe**

Sygnał	Symbol	Parametr	Min.	Max.	Jednostka
$\overline{MREQ}$	$t_{DL(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock. $\overline{MREQ}$ Low.	20	85	ns
A0 ÷ A15	$t_{D(AD)}$	Address Output Delay		110	ns

Z powyższych danych i wykresu czasowego (rys. 5) wynika, że aktywny sygnał $\overline{MREQ}$ może się pojawić najwcześniej 20 ns po opadającym zboczu zegara Φ w takcie T1, a najpóźniej 85 ns po tym zboczu. Interpretację parametru $t_{D(AD)}$ pozostawiamy czytelnikom.



Rys. 5. Wykres cyklu odczytu pamięci
(a) — ustawienie adresu, (b) — ustawienie sygnałów sterujących w stan aktywny, (c) — odczyt 8-bitowej danej z magistrali danych, (d) — ustawienie sygnałów sterujących w stan nieaktywny

Cykl zapisu pamięci

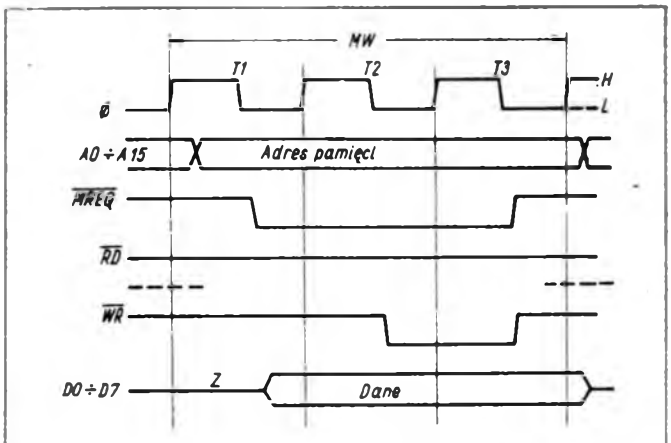
Podobnie jak przy odczycie, operacja zapisu rozpoczyna się od ustawienia 16-bitowej liczby binarnej na magistrali adresowej A0÷A15. Liczba ta jest adresem komórki pamięci, do której w takcie T3 zostanie zapisana 8-bitowa liczba binarna. Wraz z opadającym zboczem zegara w takcie T1 jest uaktywniony sygnał $\overline{MREQ}$, zaś na magistralę danych mikroprocesor wystawia 8-bitową liczbę binalną przeznaczoną do zapisu.

Opadające zbocze sygnału $\overline{MREQ}$ inicjuje poszukiwanie komórki pamięci o adresie widniejącym na magistrali adresowej. O tym czy i kiedy 8-bitowa liczba binarna ustawiona na magistrali danych zostanie wpisana do tej komórki decyduje sygnał $\overline{WR}$. Sygnał ten jest uaktywniony wraz z opadającym zboczem zegara w takcie T2.

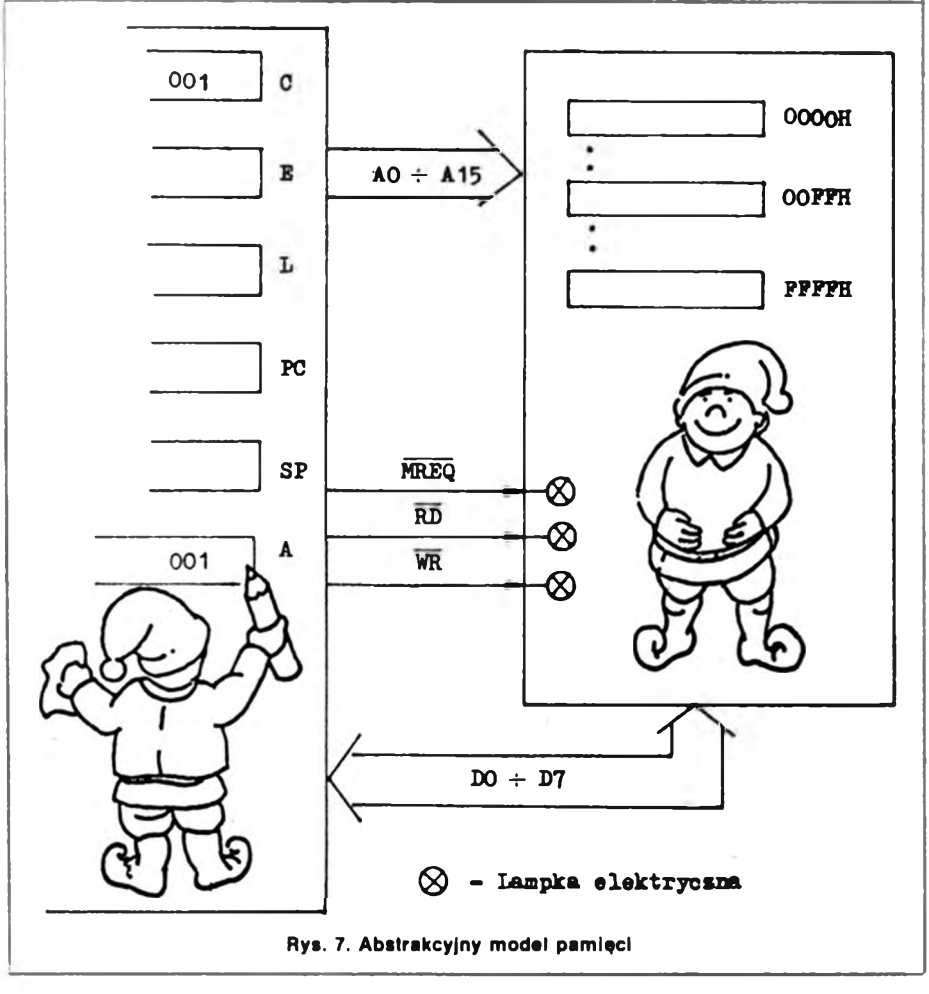
Operacja wpisywania danej do pamięci kończy się wraz z narastającym zboczem sygnału $\overline{WR}$. W tym samym czasie mikroprocesor ustawia sygnał $\overline{MREQ}$ w stan „nieaktwny" powodując odcięcie pamięci zewnętrznej od magistrali danych. Należy zauważyć, że sygnał sterujący odczytem $\overline{RD}$ pozostaje przez cały czas trwania operacji zapisu w stanie „nieaktywnym". Cykl zapisu jest przedstawiony na rys. 6.

Abstrakcyjny model pamięci

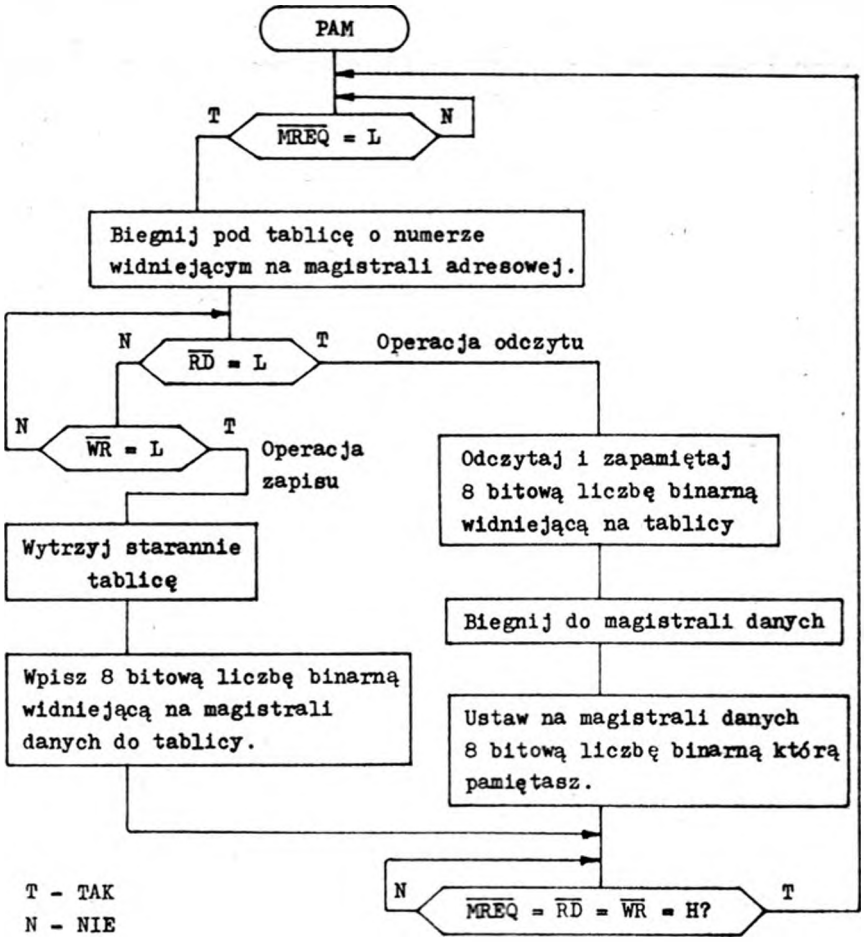
Jak widać, w pamięci zewnętrznej mikroprocesora pracuje kolega MIKa krasnoludek o imieniu PAM. Ma on do dyspozycji aż 65536 (2^{16}) małych tablic szkolnych, na których może zapisywać 8-bitowe liczby binarne. Wszystkie tablice są ponumerowane i uporządkowane dzięki czemu PAM może łatwo odszukać tablicę o dowolnym numerze. Używane w poprzednich rozdziałach określenie „KOMÓRKA PAMIĘCI" to nic innego jak tablica szkolna, zaś „ADRES KOMÓRKI PAMIĘCI", to mumer tablicy. Model pamięci jest przedstawiony na rys. 7. Współpracę MIKa i PAMa obserwowaliśmy już na wykresach czasowych. Wiemy, że komunikują się między sobą za pomocą dwóch elementarnych operacji — odczytu i zapisu. Współpracę tę obserwowaliśmy z punktu widzenia mikroprocesora. Popatrzmy teraz na to samo z punktu widzenia pamięci. Zobaczmy, jak reaguje PAM na sygnały generowane przez MIKa. W chwili wejścia do poniższego algorytmu wszystkie sygnały sterujące są nieaktywne, tzn. $\overline{MREQ} = \overline{RD} = \overline{WR} = H$. Lampka świeci się, gdy sygnał sterujący jest aktywny.



Rys. 6. Wykres czasowy cyklu zapisu pamięci



Rys. 7. Abstrakcyjny model pamięci



Rys. 8. Algorytm postępowania PAMa

Czynności, jakie musi wykonać krasnoludek PAM, przedstawiono w postaci schematu blokowego na rys. 8.

Powyższy algorytm określa wszystkie czynności jakie może wykonać krasnoludek PAM. Jak widać, jest on dobrze dostosowany do przebiegów czasowych, które generuje MIK.

Zauważmy, że PAM pracuje w sposób asynchroniczny, tzn. jego praca nie jest synchronizowana żadnym zewnętrznym zegarem. Wszystkie rozkazy MIKa stara się wykonać najszybciej jak tylko potrafi. Nowoczesne pamięci mają czas dostępu $t_{ACC} \leqslant 250$ ns, co oznacza, że pracujący w nich krasnoludek potrafi wykonać operację odczytu/zapisu na dowolnej tablicy w ciągu zaledwie 250 ns.

Cykl pobrania rozkazu

Mikroprocesor podobnie jak każdy komputer wykonuje ciąg rozkazów napisanych przez człowieka. Pobiera je rozkaz po rozkazie ze swojej pamięci zewnętrznej. Z poprzednich artykułów wiemy jednak, że człowiek zapisuje rozkazy mikroprocesora w postaci symbolicznej lub słownej.
W jaki zatem sposób człowiek komunikuje się z MIKiem? Odpowiedź jest bardzo prosta.

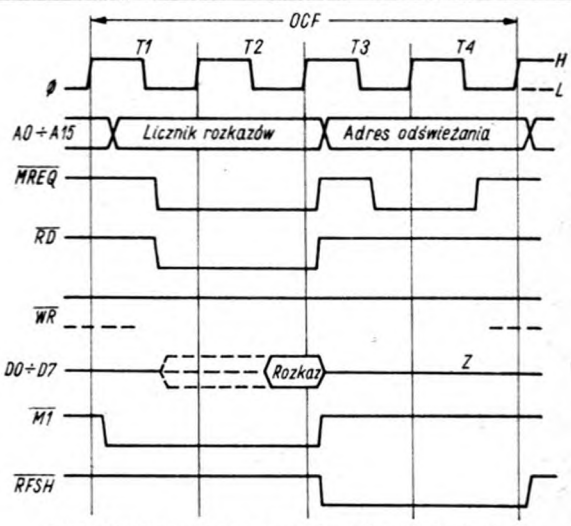
Każdy rozkaz słowny, który zna człowiek, ma swój odpowiednik w liczbach binarnych.

W tablicy 2 podano przykłady znanych nam już rozkazów mikroprocesora Z80 oraz odpowiadające im liczby binarne zrozumiałe dla MIKa. Jeśli człowiek chce, aby MIK wykonał powyższy ciąg rozkazów, musi umieścić w pamięci zewnętrznej kody rozkazów, a nie zapisy słowne lub symboliczne.

Tablica 2. **Przykłady rozkazów procesora i ich kodu maszynowego**

Zapis symboliczny	Zapis słowny	Kod rozkazu
A ——— C	LD A, C	79H
SP ←——— HL	LD SP, HL	F9H
B ←——— A	LD B, A	47H

Każdy binarny kod rozkazu jest pobierany w pamięci zewnętrznej i kierowany do środka mikroprocesora za pomocą cyklu. Cykl pobrania kodu rozkazu jest oznaczany OCF (ang. op. code fetch). Wykres czasowy cyklu pobrania kodu przedstawiono na rys. 9.
Jeżeli do wykonania pobranego rozkazu jest konieczny odczyt danej (8-bitowa liczba binarna) z pamięci, mikroprocesor pobiera tę daną generując cykl MR. Jeżeli do wykonania pobranego rozkazu jest konieczny zapis danej do pamięci, mikroprocesor zapisuje tę daną za pomocą cyklu MW.
Pierwsze dwa takty powyższego wykresu są bardzo podobne do wykresu czasowego MR. Na początku taktu T1 jest ustawiany adres komórki pamięci, zaś wraz z opadającym zboczem zegara uaktywniane sa sygnały $\overline{MREQ}$ = = $\overline{RD}$ = L. Mikroprocesor odczytuje kod rozkazu wraz z narastającym zboczem zegara na początku taktu T3 sprowadzając jednocześnie sygnały $\overline{MREQ}$, $\overline{RD}$ do stanu nieaktywnego.
W takcie T3 i T4 jest wykonywana operacja tzw. odświeżania pamięci dynamicznych. Polega ona na tym, że na siedmiu mniej znaczących bitach magistrali adresowej (A0÷A6) jest usta-



Rys. 9. Wykres czasowy cyklu pobrania instrukcji

wiana liczba binarna będąca adresem odświeżania. Za każdym razem, kiedy mikroprocesor wykonuje cykl OCF, adres odświeżania jest zwiększany o 1. Operacją odświeżania sterują sygnały $\overline{MREQ}$, $\overline{RFSH}$. Pamięciami dynamicznymi i operacją odświeżania nie będziemy się zajmować, gdyż nie są one stosowane w małych systemach mikroprocesorowych. Zauważmy, że w cyklu OCF na magistralę adresową jest wystawiana zawsze zawartość 16-bitowego rejestru zwanego LICZNIKIEM ROZKAZÓW (ang. program counter) i oznaczanego PC. W końcu wykonywania każdego rozkazu licznik rozkazów wskazuje zawsze następny rozkaz do wykonania. Rozkaz ten jest pobierany oczywiście, za pomocą cyklu OCF. Mikroprocesor sygnalizuje „światu zewnętrznemu" operację pobierania kodu rozkazu za pomocą specjalnej linii wyjściowej $\overline{M1}$.

Sygnał $\overline{M1}$ jest aktywny zawsze wtedy, gdy mikroprocesor pobiera kod rozkazu

Sygnał $\overline{M1}$ jest nieaktywny w znanych nam cyklach MR i MW □

mgr inż. Stanisław Gardynik

Podstawy techniki mikroprocesorowej (6)

Współpraca mikroprocesora z układami wejścia/wyjścia

Wiemy już, w jaki sposób mikroprocesor współpracuje z pamięcią zewnętrzną. Nie wiemy jednak nic na temat współpracy mikroprocesora z urządzeniami zewnętrznymi, takimi jak: klawiatura, drukarka.

Mikroprocesor komunikuje się z urządzeniami zewnętrznymi za pomocą układów wejścia/wyjścia zwanych portami.

Abstrakcyjny model bloku portów wejściowych oraz bloku portów wyjściowych z punktu widzenia procesora jest analogiczny do przedstawionego w poprzednim odcinku abstrakcyjnego modelu pamięci. Najważniejsza różnica polega na tym, że zawartość każdego z rejestrów wewnętrznych jest dodatkowo dostępna z zewnętrznego „świata człowieka" za pośrednictwem ośmiu linii we/wy.

W przypadku procesora Z80 porty wejścia/wyjścia są adresowane za pomocą mniej znaczącego bajtu magistrali adresowej, czyli można tu podłączyć co najwyżej 256 portów wejściowych i 256 portów wyjściowych. W praktyce, najczęściej wykorzystuje się zaledwie kilka portów wejściowych i kilka wyjściowych.

Mikroprocesor z pamięcią zewnętrzną i portami wejścia/wyjścia tworzy system mikroprocesorowy, którego schemat blokowy przedstawiono na rys. 1.

Mikroprocesor komunikując się z portem wejściowym realizuje tzw. cykl odczytu we/wy — PR, a komunikując się z portem wyjściowym cykl zapisu we/wy — PW.

Procesor może odczytać informację z układów wejściowych, gdy sygnały sterujące są w następujących stanach:

$\overline{MREQ} = H$ $\overline{IORQ} = L$ $\overline{WR} = H$ $\overline{RD} = L$

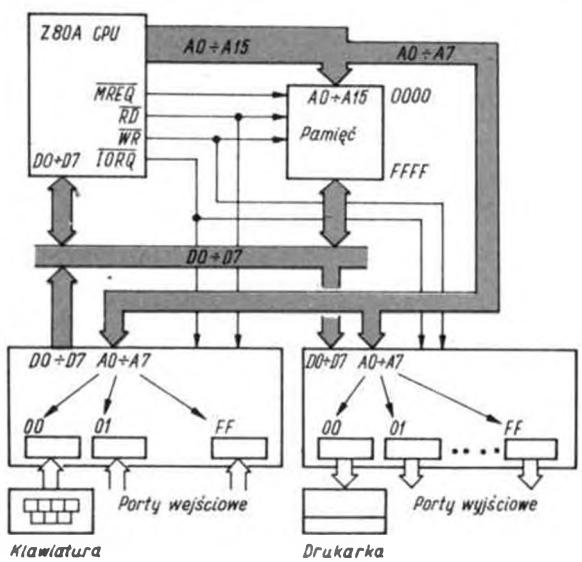
zaś zapisać informację do portów wyjściowych:

$\overline{MREQ} = H$ $\overline{IORQ} = L$ $\overline{WR} = L$ $\overline{RD} = H$

Komunikacja mikroprocesora z portami wejścia/wyjścia jest realizowana analogicznie do znanej nam już komunikacji mikroprocesora z pamięcią. Najważniejsza różnica polega na tym, że sygnał żądania dostępu do pamięci — $\overline{MREQ}$ jest zastąpiony sygnałem żądania dostępu do portów wejścia/wyjścia — $\overline{IORQ}$.

W czasie, gdy mikroprocesor komunikuje się z pamięcią (cykle MR, MW, OCF), sygnał $\overline{IORQ}$ jest w stanie nieaktywnym powodując odcięcie portów wejścia/wyjścia od magistrali danych. Analogicznie, w czasie gdy mikroprocesor komunikuje się z portami wejści/wyjścia (cykle PR, PW), sygnał $\overline{MREQ}$ jest w stanie nieaktywnym powodując odcięcie pamięci od magistrali danych.

W czasie komunikacji z portami wejściowymi ($\overline{IORQ}$ = L i $\overline{RD}$ = L) sygnał $\overline{WR}$ = H wymusza odcięcie bloku portów wyjściowych. W czasie komunikacji z portami wyjściowymi ($\overline{IORQ}$ = L i $\overline{WR}$ = L) sygnał $\overline{RD}$ = H wymusza odcięcie bloku portów wejściowych.



Rys. 1. Schemat blokowy systemu mikroprocesorowego

$\overline{IORQ}$ — sygnał żądania dostępu do urządzeń we/wy

$\overline{RD}$ — sygnał sterujący odczytem

$\overline{WR}$ — sygnał sterujący zapisem

Cykl odczytu portu wejściowego

Przeanalizujmy teraz na przykładzie dwóch modeli portów wejściowych działanie procesora przy ich odczycie. Pierwszy model przedstawiony na rys. 2, to bufor wejściowy trójstanowy, najprostszy typ układu wejściowego. Zakładając, że nasz krasnoludek MIK steruje przełącznikiem EW zgodnie z tablicą stwierdzamy, że elektroniczne wyłączniki EW zwierają wtedy i tylko wtedy, gdy

$$AO \div A7 = N \qquad \overline{IORQ} = L \qquad \overline{RD} = L$$

Sposób działania układu wejściowego typu bufor trójstanowy

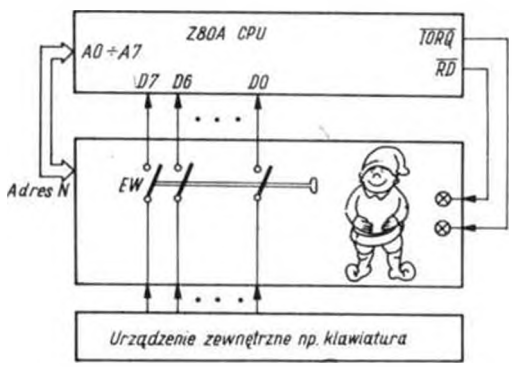
$\overline{IORQ}$	$\overline{RD}$	Adres AO ÷ A7	Wyłączniki EW
L	L	N	Włączone
L	L	różny od N	Wyłączone
L	H	X	Wyłączone
H	X	X	Wyłączone

Informacja z urządzenia zewnętrznego jest wówczas bezpośrednio podawana na magistralę danych procesora i każda z linii wejściowych jest w jednym z dwóch stanów H lub L. W przeciwnym wypadku bufor jest w stanie wysokiej impedancji — rozwarcia tzw. staniem trzecim.

Drugi model, to port wejściowy z rejestrem wejściowym (rys. 3), wymaga bardziej szczegółowej analizy.

Mikroprocesor odczytuje stan portu wejściowego o adresie N generując sygnały:

AN ÷ A7 = N i $\overline{IORQ}$ = L i $\overline{RD}$ = L (cykl maszynowy PR)



Rys. 2. Model portu wejściowego typu „bufor trójstanowy"

RWE — rejestr wejściowy (tablica szkolna)

STB — sygnał wpisujący daną ustawioną przez urządzenie zewnętrzne do rejestru RWE

STB = L — dana jest przepisywana

STB = H — zabronione przepisywania

IBF — sygnał informujący o stanie rejestru RWE

IBF = H — rejestr wejściowy pełny, tzn. dana została wpisana do rejestru RWE, lecz mikroprocesor jeszcze jej nie odczytał.

IBF = L — rejestr wejściowy pusty, tzn. mikroprocesor odczytał daną ustawioną w rejestrze RWE.

Urządzenie zewnętrzne może wpisać następną daną (za pomocą sygnału STB).

INTR — sygnał zgłoszenia przerwania

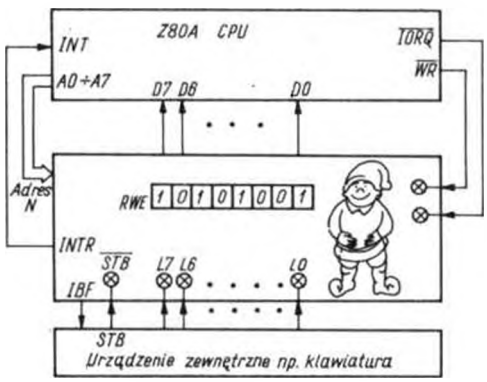
INTR = L — brak zgłoszenia

INTR = H — zgłoszenie przerwania. Port wejściowy zgłasza przerwanie, gdy rejestr wejściowy jest pełny (IBF = H) i sygnał STB zmienił swój stan z „L" na „H".

INT — wejście przerywające mikroprocesora

INT = H — zgłoszenie przerwania

INT = L — brak zgłoszenia przerwania



Rys. 3. Model portu wejściowego typu „rejestr wejściowy"

Mechanizm komunikacji urządzenia zewnętrznego z mikroprocesorem jest w tym wypadku następujący.
Mikroprocesor odczytując daną z rejestru RWE powoduje ustawienie

IBF = L — rejestr wejściowy pusty

Urządzenie zewnętrzne widząc LBF = L, ustawia na liniach LO ÷ L7 następną daną dla mikroprocesora, po czym uaktywnia sygnał $\overline{STB}$.
Stan $\overline{STB}$ = L jest sygnałem dla krasnoludka, aby natychmiast przystąpił do przepisywania stanu wejść LO ÷ L7 do rejestru RWE. Po zakończeniu przepisywania krasnoludek ustawia

IBF = G — bufor wejściowy pełny

Urządzenie zewnętrzne reaguje na stan IBF = H sprowadzając sygnał $\overline{STB}$ do stanu nieaktywnego ($\overline{STB}$ = H).
Krasnoludek, widząc zmianę sygnału $\overline{STB}$ z „L" na „H" oraz IBF = H generuje przerwanie do mikroprocesora ustawiając

INTR = H — zgłoszenie przerwania

Pracujący w mikroprocesorze MIK, obojętnie co by w tym czasie nie robił, przerwie swoją pracę i odczyta stan portu wejściowego, zgłaszającego przerwanie. Po odczytaniu danej z rejestru RWE stan linii IBF i INTR będzie następujący:

IBF = L — rejestr wejściowy pusty

INTR = L — brak zgłoszenia przerwania

Zmiana sygnału IBF z „H" na „L" jest sygnałem dla urządzenia zewnętrznego, że może przesłać następną daną.
Zauważmy, że urządzenie zewnętrzne nie powinno przesyłać danej w chwili, gdy IBF = H (rejestr wejściowy pełny), gdyż zniszczy poprzednią daną, której mikroprocesor nie zdążył jeszcze odczytać.

Powyższy sposób komunikacji urządzenia zewnętrznego z mikroprocesorem nosi nazwę „przesłanie danej z potwierdzeniem" (ang. handshake). Sygnały sterujące realizujące ów „handshaking", to oczywiście $\overline{STB}$ i IBF.
Sposób współpracy procesora z układami wejściowymi przedstawiono na wykresie czasowym cyklu odczytu PR — rys. 4.
Jak widać, na początku taktu T1 mikroprocesor ustawia adres portu wejściowego, z którego chce odczytać informację. Na początku taktu T2 uaktywnione zostają sygnały sterujące odczytem. W bloku portów wejściowych rozpoczyna się wówczas gorączkowe poszukiwanie portu o adresie ustawionym na A0 ÷ A7, bowiem stabilna informacja z tego portu MUSI się pojawić najpóźniej tuż przed opadającym zboczem zegara w takcie T3. Mikroprocesor odczytuje wówczas stan magistrali danych. Chwilę po odczytaniu danej, sygnały sterujące odczytem sprowadzane są do stanu nieaktywnego odcinając port wejściowy od magistrali danych. Zauważmy, że dzięki wstawieniu taktu T_W^* do każdego cyklu PR mikroprocesor Z80 może współpracować z wolnymi portami wejściowymi o czasie dostępu:

$$T_{ACC} = 2.5T = 2.5 \cdot 250 = 625 \text{ nS (dla CA80)}$$

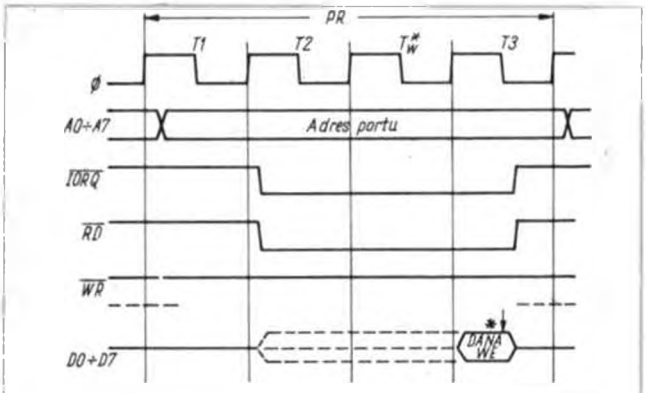
Cykl zapisu portu wyjściowego

Rozpatrzmy teraz zachowanie się procesora przy współpracy z modelowym układem wyjściowym z rys. 5.
Mikroprocesor wpisując daną do portu wyjściowego powoduje uaktywnienie sygnału $\overline{OBF}$. Stan $\overline{OBF}$ = L informuje urządzenie zewnętrzne, że rejestr wyjściowy jest pełny i można go odczytać.
Urządzenie zewnętrzne odczytuje stan rejestru RWY uaktywniając linię $\overline{ACK}$. Stan $\overline{ACK}$ = L (podczytywanie danej) wymusza przejście sygnału $\overline{OBF}$ do stanu nieaktywnego ($\overline{OBF}$ = H). Po odczytaniu danej urządzenie zewnętrzne sprowadza sygnał $\overline{ACK}$ do stanu nieaktywnego ($\overline{ACK}$ = H). Pracujący w porcie wyjściowym krasnoludek widząc zmianę sygnału $\overline{ACK}$ z „L" na „H" oraz $\overline{OBF}$ = H (RWY — pusty) uaktywnia linię INTR = H) żądając od mikroprocesora przysłania następnej danej dla urządzenia zewnętrznego.
Mikroprocesor wpisując daną do rejestru RWY (cykl maszynowy PW) wymusza następujące stany sygnałów INTR i $\overline{OBF}$.

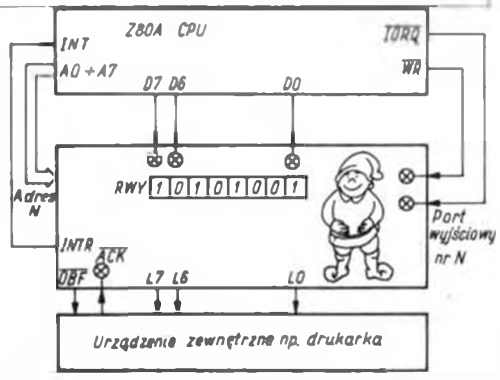
INTR = L — brak żądania przerwania

$\overline{OBF}$ = L — rejestr wyjściowy pełny

Urządzenie zewnętrzne widząc $\overline{OBF}$ = L odczytuje zawartość rejestru wyjściowego uaktywniając sygnał $\overline{ACK}$ .... itd.



Rys. 4. Cykl odczytu portu wejściowego PR (ang. port read)
T_W^* — takt oczekiwania wstawiany przez mikroprocesor w każdym cyklu PR
*↓ — moment odczytu magistrali danych.

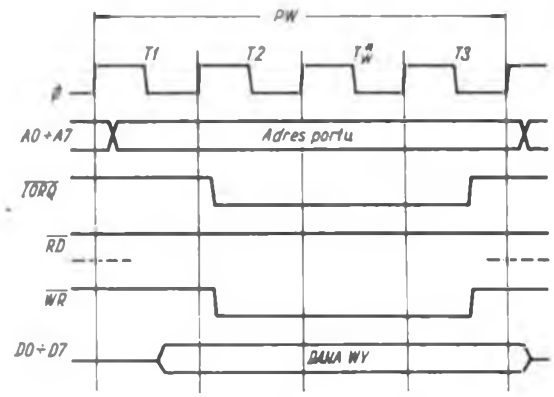


Rys. 5. Model portu wyjściowego typu rejestr wyjściowy

$\overline{RWY}$ — rejestr wyjściowy (tablica szkolna)
ACK — sygnał potwierdzający odczytywanie danej przez urządzenie zewnętrzne
$\overline{ACK}$ = L — urządzenie zewnętrzne odczytuje daną z rejestru wyjściowego RWY
$\overline{ACK}$ = zmiana z „L" na „H" — koniec odczytywania danej
$\overline{OBF}$ — sygnał informujący urządzenie zewnętrzne o stanie rejestru RWY
$\overline{OBF}$ = L — rejestr wyjściowy pełny, tzn. mikroprocesor wpisał daną do rejestru RWY, lecz urządzenie zewnętrzne jeszcze jej nie odczytało
$\overline{OBF}$ = H — rejestr wyjściowy pusty, tzn. urządzenie zewnętrzne odczytało daną z rejestru RWY. Mikroprocesor może wpisać następną daną
INTR — sygnał zgłoszenia przerwania
INTR = L — brak zgłoszenia
INTR = H — zgłoszenie przerwania. Port wyjściowy zgłasza przerwanie, gdy rejestr wyjściowy jest pusty ($\overline{OBF}$ = H) i sygnał $\overline{ACK}$ zmienił swój stan z „L" na „H"
INT — wejście przerywające mikroprocesora
INT = H — zgłoszenie przerwania
INT = L — brak zgłoszenia

Wykres czasowy cyklu zapisu przedstawiono na rysunku 6. Na początku taktu T1 mikroprocesor ustawia adres portu wyjściowego, zaś w jego środku stabilną daną wyjściową. sygnały sterujące ($\overline{IORQ}$ i $\overline{WR}$) są uaktywniane na początku taktu T2. Ostateczny wpis informacji do portu wyjściowego następuje wraz z narastającym zboczem sygnału $\overline{WR}$ w połowie taktu T3.

Tym odcinkiem kończymy pierwszą część cyklu „Podstawy techniki mikroprocesorowej". Osoby zainteresowane formą wykładu wprowadzającego w tajniki mikroelektroniki, prezentowanego przez Pana Stanisława Gardynika, odsyłamy do jego firmy „MIK" 05-090 Raszyn, ul. Olszowa 68, gdzie mogą nabyć kolejne tomiki jego książek oraz płytki i elementy do samodzielnego montażu sterownika CA 80. Na naszych łamach wrócimy do tematu w przyszłości, o ile nasi młodzi czytelnicy wyrażą zainteresowanie taką żartobliwą formą artykułów. Oczekujemy również sugestii dotyczących tematyki kolejnych cykli w Klubie Młodych Elektroników. □



Rys. 6. Cykl zapisu portu wyjściowego PW (ang port write)
T_W^* — takt oczekiwania wstawiany przez mikroprocesor w każdym cyklu PW.