



Był początek lat siedemdziesiątych, gdy mnie, małoletniemu pacholęciu, wpadł do ręki numer bodaj "Horyzontów Techniki". Jeden z artykułów był poświęcony mi.. mi..., o! mikroprocesorowi. Nie pamiętam treści artykułu, zapamiętałem natomiast rysunkowy żart stanowiący doń ilustrację. Otóż dwóch naukowców - poważnych panów ze staromodnymi bródkami, przechadzając się po parku, natyka się na kopiaсте mrowisko, z którego startuje rakietą kosmiczną. Jeden z nich wykrzykuje do drugiego: już wiem, gdzie zgubiłem swój mikrokomputer!

Od tamtego zdarzenia minęło może dwadzieścia kilka lat, a ten żart pomalutka zaczyna się materializować - dziś, może nie mrówki, ale bardzo młodzi ludzie mogą realizować projekty o wręcz kosmicznym poziomie zaawansowania - a wszystko to za sprawą niewielkiej krzemowej kostki, w której zaklęto metody i sposoby odpowiedzi na zadawane jej pytania. Niegdyś nazywano komputery mózżami elektronowymi - taka nazwa od początku budziła szacunek i podziw - niektórych wręcz rzucała na kolana. Jednak do mózżu, tego, który przychodzi nam nosić w głowie, bardzo mu daleko.

Tymczasem w mikroprocesorach tak naprawdę nie ma nic tajemniczego. Zrozumieć

działanie mikroprocesora, a nawet go zaprogramować, może dosłownie każdy. Nie potrzeba do tego wielkiej wiedzy - trzeba natomiast nieco *przestawić swój sposób myślenia*.

Chcę tu uzmysłowić na przykładzie: codziennie rano budzisz się ze snu i ubierasz się. To oczywiste i proste - zupełnie się nie zastanawiasz, co masz po kolei zrobić, nawet, gdy jesteś zaspany po "rozrywkowej" nocy, prawie nigdy nie zdarza ci się założyć koszuli na sweter. Pomyśl jednak - wykonujesz pewien ciąg elementarnych czynności, czyli realizujesz pewien *program* zapisany w twojej pamięci. Ty się tego nauczyłeś i o tym pamiętasz; nie musisz już zatrudniać swojej świadomości. W czasie wykonywania tego programu pewne kroki realizujesz bezwarunkowo, podejmujesz też decyzje zależne od warunków zewnętrznych, poza tym niejednokrotnie realizujesz taki sam ciąg czynności - nazwijmy go podprogramem lub procedurą.

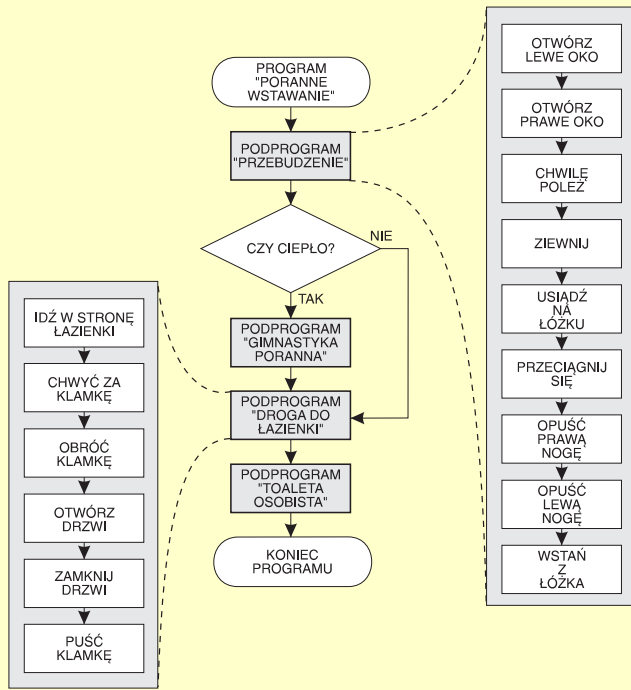
Rozłóżmy więc twoje poranne wstawanie na składniki: otwierasz oczy, chwilę (?) leżysz, ziewasz, siadasz na łóżku, przeciągasz się, opuszczasz... oczywiście prawą nogę, opuszczasz lewą nogę, wstajesz, znowu ziewasz, zastanawiasz się czy się trochę nie pogimnasty-

kować - podejmujesz jednak decyzję: NIE, bo jest zbyt zimno. Pomijasz podprogram "gimnastyka poranna" i realizujesz podprogram "droga do łazienki", na który składają się kroki w kierunku łazienki, chwycenie klamki, otwarcie drzwi, wejście, zamknięcie drzwi. Gdy jesteś w łazience nadal wykonujesz mnóstwo elementarnych czynności, które w sumie składają się na dość duży program pt. "poranne wstawanie" (rys. 1).

I tu pokazuję ci klucz do zrozumienia mikroprocesora: jego działanie polega na wykonywaniu wielu naprawdę prostych czynności. Okazuje się bowiem, że każde, nawet najbardziej skomplikowane zadanie można przedstawić jako złożenie pewnej liczby bardzo prostych kroków. Z kolei mikroprocesor czy mikrokomputer potrafi wykonywać tylko bardzo proste czynności i obliczenia, które nazywamy rozkazami. Daleko mu więc do mózżu, za to jest piorunisko szybki.

Programowanie polega więc na zapisaniu ciągu czynności, jakie potrafi mikroprocesor wykonać, a takie czynności nazywane są *rozkazami* albo *instrukcjami*.

Mikroprocesor to maszyna, która wykonuje rozkazy. Pobiera rozkaz i natychmiast przystę-



Rys. 1. Treść zapisów może śmieszyć, jednak i tu zawiera się głębsza myśl. Te czwotkąty tworzą algorytm, który jest graficznym przedstawieniem toku postępowania w celu wykonania jakiegoś zadania. W prostokątach zawieramy pewne czynności, które uznajemy już za niepodzielne na danym poziomie rozumowania. Gdy jednak chcemy je uszczegółowić, wytwarzamy na boku kolejny ciąg czynności, jeszcze bardziej elementarnych. Proces taki nazwiemy zagnieżdżaniem algorytmu, a metoda układania algorytmu w taki sposób jest nazywana metodą zstępującą. Najpierw układamy algorytm ogólny, który przedstawia kilka bloków funkcjonalnych, a potem każdy z nich zaczynamy rozpisywać na coraz prostsze operacje. Ten proces zagnieżdżania możemy prowadzić w nieskończoność. Dla naszego przykładu: opisujemy naciśnięcie klamki poprzez skurcze kolejno uruchamianych mięśni. Zapisz więc, w ramach ćwiczenia mikroprocesorowego myślenia, ciąg czynności dla podprogramu "Gimnastyka poranna" i "Toaleta osobista". Jeśli zrobisz to uczciwie, z kilkoma poziomami zagnieżdżeń, to zobaczysz, że kartka papieru będzie za mała dla pomieszczenia całego algorytmu. Przy okazji poznajemy umowne znaki graficzne, służące do zapisu elementów algorytmu. Realizowane czynności umieszczamy w prostokątach, w owalach przyjęło się zapisywać początek i koniec algorytmu, przy czym w owalu początkowym piszemy, czego dotyczy ten algorytm. To później nam ułatwi analizę poprawności działania algorytmu i odszukiwanie powiązań z innymi algorytmami. Przejścia pomiędzy blokami, narysowane strzałkami, tworzą ścieżkę logiczną programu. Interesującym blokiem jest romb - bloki warunkowego rozgałęzienia ścieżki logicznej programu. Sformułowanie weń wstawiane jest przeważnie pytaniem, na które można odpowiedzieć dwojako: tak albo nie. Zależnie od odpowiedzi program obiera jedną z dróg dalszej realizacji.

puje do jego wykonania. Skąd pobiera? Z pamięci! W każdym systemie mikroprocesorowym musi być jakaś pamięć. My też pamiętamy o każdej czynności, jeśli idziemy do łazienki, ale tego sobie nie uświadamiamy.

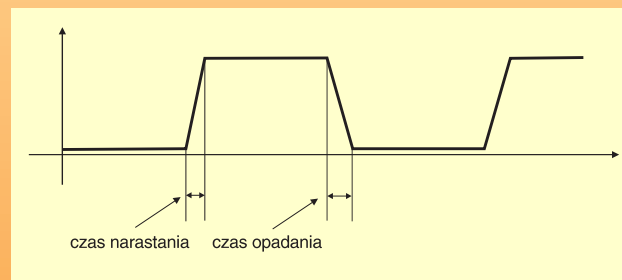
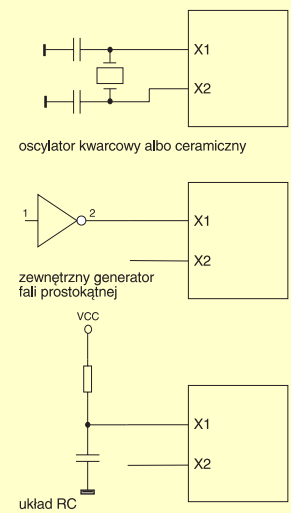
No tak, ale system zawierający "goly" mikroprocesor i pamięć pracowałby sam dla siebie, czyli poza zużyciem energii potrzebnej do realizacji rozkazów nic innego nie potrafiłby zrobić. Trzeba więc dodać jakieś usta, uszy: nazwijmy je wrotami lub z angielska - portem. Za pomocą portów kostka mikroprocesora kontaktuje się ze światem zewnętrznym. Wpiszmy jeszcze, lub mówiąc mikroprocesorową nowomową - zaimplementujmy do mikroprocesora metody komunikacji z różnego rodzaju portami.

I tak oto stworzyliśmy mikrokomputer! I jeśli nawet piejesz z zachwytu nad możliwościami współczesnych mikrokomputerów, to pamiętaj, że mikrokomputer zawsze składa się z mikroprocesora, pamięci i portów. Resztę stanowią dodatki "uszlachetniające" cały wyrób.

Ten wstęp ma pokazać, że mikroprocesor jest maszyną niewiele bardziej skomplikowaną od tokarki (tokarka ma uchwyt, suport i łożo, a reszta to dodatki "uszlachetniające" wyrób) czy samochodu (samochód ma koła, hamulce i silnik, a reszta to dodatki "uszlachetniające" wyrób). Mikroprocesor - to ta straszna nazwa powoduje, że wielu zapalonych elektroników nie sięga po rozwiązywanie mikroprocesorowe, pozostając przy tradycyjnym rozwiązaniu hardware'owym. Wynika to ze strachu przed tabu. A to jest zwykła zabawka, no może trochę narowista, ale jednak zabawka, jedna z tych, jakimi nam, dzieciom cywilizacji końca XX wieku, przyszło się bawić. W swojej zawodowej pracy tak uzależniłem się od mikroprocesorów, że nawet proste funkcje wolę zrobić na malutkim procesorku insektowym niż kombinować z układem bramek, przerzutników i liczników. Już przy niewielkiej wprawie w posługiwaniu się mikroprocesorami ich "giętkość", łatwość zmiany ca-

Sygnal zegarowy może być wytworzony w samej kostce mikroprocesora, jest to cecha większości mikrokontrolerów jednokładowych. Producent takiego układu wydziela wtedy końcówki, do których można podłączyć rezonator kwarcowy czy ceramiczny. Te same końcówki mogą też przyjmować sygnał zegarowy spoza układu, z innego, zewnętrznego generatora. Rezonator kwarcowy daje sygnał o dużej stabilności, nieco gorszą stabilnością charakteryzuje się układ z rezonatorem ceramicznym. Kiedy zależy nam na precyzyjnym odmierzeniu czasu, z pewnością zegar zbudowany na kwarcu ma uzasadnienie. Jednak zegar może mieć częstotliwość znaną tylko orientacyjnie, np. układ może wyłączać lampkę na biurku po dwóch minutach $\pm 3s$. Wystarczy więc w miejsce kwarcu wluć prosty układ RC. Współczesne mikrokontrolery jednokładowe to zapewniają, ale po szczegóły musimy sięgnąć do katalogu. Na rysunku pokazano trzy typowe sposoby podłączenia oscylatorów do mikrokomputera jednokładowego.

Bardziej skomplikowane mikroprocesory, szczególnie te 16-bitowe i silniejsze, wymagają osobnego układu zegara. Nierzadko taki zegar musi spełniać ostrzejsze wymagania niż to było w przypadku mikrokontrolerów jednokładowych, a szczególnie zwraca się w nim baczniejszą uwagę na stabilność częstotliwości sygnału, dopuszczalne nachylenia zboczy generowanej fali prostokątnej. Musimy wiedzieć, że sygnał fali prostokątnej występuje tylko w teorii, to ideał, w praktyce zawsze dopatrzmy się pewnego czasu potrzebnego na zmianę



nę poziomu napięcia z niskiego na wysoki i odwrotnie. Czas potrzebny na zmianę z poziomu niskiego na wysoki nazywamy czasem narastania (rise time), zaś czas potrzebny na zmianę sygnału z poziomu wysokiego na niski to czas opadania (fall time). Oba te czasy z-

na są też pod wspólną nazwą czasów trwania zboczy. Im te czasy są dłuższe, to mówimy, że zbocza posiadają coraz mniejsze nachylenie i fala prostokątna coraz bardziej przypomina przebieg trapezowy.

Podajmy skrajny przykład. Układy zegarowe produkowane z myślą o procesorze Pentium muszą zapewnić sygnał zegarowy o czasach trwania zboczy rzędu kilkudziesięciu ps! (1ps = 1 pikosekunda = $10^{-12}s$ = 0.000001μs, 1μs trwa okres przebiegu o częstotliwości 1MHz)

Jeszcze nie tak dawno, kiedy mikroprocesory posiadały słowo co najwyżej 8-bitowe, obok nich spokojnie egzystowały układy mikroprogramowane jako osobne układy scalone. Układ mikroprogramowany wymaga zewnętrznej pamięci ROM, w której były zapisane mikroinstrukcje, tworzące mikroprogram. Taka mikroinstrukcja zawierała kod bardzo prostej czynności, z reguły były to sygnały zezwolenia na te czynności, ewentualny adres następnego rozkazu, czasem kilka bitów reprezentujących stany warunkowe. Z chwilą, gdy mikroprocesory stawały się coraz tańsze i gwałtownie zaczęły rosnać ich możliwości, zaniechano produkcji pojedynczych układów mikroprogramowanych. Układy mikroprogramowane nadal są, stały się fragmentem... mikroprocesorów. Właściwie bez nich nie można byłoby dekodować instrukcji przychodzących do mikroprocesora. Pamięć mikroprogramu też została scalona w strukturze mikroprocesora i teraz cały układ mikroprogramowany nazwiemy dekodorem rozkazów.

go rozwiązania na inne, staje się zaletą nie do odrzucenia.

Celem niniejszego artykułu nie jest przedstawienie dokładnego opisu konkretnego typu mikroprocesora, temu służą katalogi firmowe, lecz ukazanie Czytelnikowi podstawowego słownictwa oraz mikroprocesorowego elementarza. Dla zrozumienia materiału wystarczy podstawowa wiedza z zakresu techniki cyfrowej, a ponadto trochę uwagi i wyobraźni.

Zacznijmy od tego, co jest motorem działania każdego mikroprocesora. Jest nim zegar.

Zegar

A zatem, jako się rzekło, mikroprocesor pobiera rozkazy i je wykonuje. Kiedy na dużym, XVIII-wiecznym żaglowcu kapitan dawał rozkaz do podniesienia żagli, to oznaczało, że trzeba było wciągnąć na określoną wysokość kilkaset kilogramów płótna. Żeby było trudniej, żagle były podnoszone w ściśle określonej kolejności. Nie robi tego jeden człowiek, ale grupa ludzi to potrafi. Niezmiernie ważną instytucją na żaglowcu był szantimen. Człowiek ten zajmował się śpiewaniem prostych, rytmicznych pieśni, przy czym tematyka ich nie była tak ważna, jak właśnie ich rytmiczność. Marynarze odpowiadali szantimelowi refrenem, jednocześnie zgrywając swoje wysiłki. Jednocześnie tych działań nazywamy *synchronizacją*.

Odpowiednikiem takiego szantimena jest w mikroprocesorze generator impulsów synchronizujących, który nazwano *generatorem zegarowym* lub krócej *zegarem*. Ma on wiele wspólnego ze znanym wszystkim zegarkiem naręcznym lub zegarem ściennym. Swoim regularnym "tykaniem" wyznacza on początki okresów czasu, które są przeznaczone do wykonania określonych operacji.

Szybciej "tykający" zegar to szybsze wykonywanie operacji. Stąd już prosta droga do uzasadnionych zachwytów nad IBM PC z mikropro-

cesorem Pentium taktowanym zegarem o częstotliwości 100 czy 150MHz. Mikroprocesor 8086, czyli ten, który zaczynał tę rodzinę procesorów, był "napędzany" zegarem 4-6MHz. W dalszym ciągu nie zmienia to faktu, że zegar dalej służy do zapewnienia właściwej kolejności zaprogramowanych czynności.

Dekoder rozkazów

Powtórzmy: mikroprocesor pobiera rozkazy i natychmiast je wykonuje. Jak bosman wrzeszczy na swoich marynarzy i oni rozumieją co do nich mówi, tak mikroprocesor musi wiedzieć, jak przetłumaczyć ciąg zer i jedynek przychodzących jako rozkaz na serię pojedynczych czynności. Do tego służy dekodek rozkazów.

Zanim pojedynczy rozkaz - np. komenda podniesienia jakiegoś żagla wydana przez kapitana dotarła do marynarzy, po drodze przechodziła przez bosmana. Rozkładał on całą operację podnoszenia danego żagla na operacje jeszcze drobniejsze.

Podobnie jest w mikroprocesorze: na każdy rozkaz składa się kilka jeszcze bardziej elementarnych czynności - nazywamy je *mikrooperacjami* albo *mikrorozkazami*.

Wszystkie rozkazy mikroprocesora pobierane z pamięci są rozpoznawane przez układ zwany dekodorem rozkazów. Odpowiada on, niczym bosman na statku, za właściwą kolejność wykonania mikrooperacji. Wykonanie rozkazu wymaga kilku, kilkunastu, bywa, że kilkudziesięciu okresów zegara, czyli żeby żagiel został podniesiony, nasz szantimen musi odśpiewać ileś zwrotek piosenki, czasem kilka piosenek.

Każdy cykl rozkazowy, znany także jako cykl maszynowy albo cykl procesora składa się z kilku taktów zegara. Długość cyklu maszynowego liczona taktami zegara jest dla konkretnego typu mikroprocesora stała.

Mamy więc motorek, jakim jest zegar, jest układ, który odpowiada za poprawną realizację

instrukcji, pora więc na coś, co będzie wykonywać instrukcje, tak pracowicie zdekodowane przez naszego "bosmana". Pora na "marynarzy", czyli elementy wykonawcze. Jednym z nich jest jednostka arytmetyczno-logiczna.

Jednostka arytmetyczno-logiczna

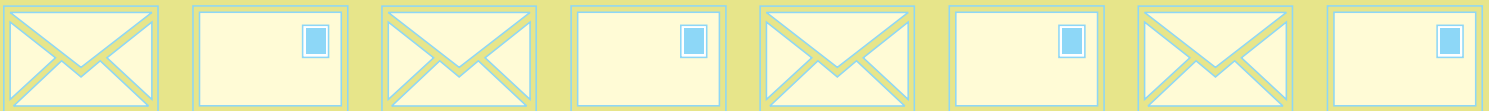
Kapitan każdego statku zawsze ma potrzebę wiedzy o swoim położeniu. Do tego celu za trudnia on nawigatora, który dba nie tylko o prawidłowe obliczenie obecnej pozycji statku, ale również wykreśla kurs, przelicza czas osiągnięcia celu przy danej prędkości itd. Pracę nawigatora można było szczegółowo przedstawić na filmie "Polowanie na Czerwony Październik", kiedy okręt płynął przez podwodny korytarz według kursu zmienianego co do sekundy.

I u nas też jest potrzebny taki rachmistrz. Rolę rachmistrza w mikroprocesorze pełni jednostka arytmetyczno-logiczna ALU (*Arithmetic Logic Unit*). Wykonuje ona wszystkie operacje arytmetyczne (dodawanie, odejmowanie, mnożenie, dzielenie) oraz logiczne (suma logiczna, iloczyn logiczny, przesunięcia bitowe). Zasadniczo jednostka arytmetyczno-logiczna jest układem niezależnym od zegara. Zawiera ona w sobie sumator, subtraktor (układ odejmujący), multiplikator (układ mnożący), układ dzielący, układ do wykonywania operacji logicznych i rejestr przesuwany. W prostych mikroprocesorach, zwłaszcza współczesnych małych mikrokontrolerach jednoukładowych układ mnożący i dzielący mogą być nieobecne, dlatego w nich mnożenie i dzielenie musi być realizowane programowo.

Jednostką ALU zawiaduje dekodek rozkazów, który najpierw ustawia rodzaj operacji, wprowadza argument bądź argumenty operacji na jej wejście, a potem odbiera wynik na wyjściu.

Z jednostką arytmetyczno-logiczną jest ściśle związany element pamiętający, który stanowi ciąg przerzutników, a nazywany jest rejestrem. Rejestr roboczy jednostki arytmetyczno-logicznej nazywany jest akumulatorem. Wartość akumulatora jest jednym z argumentów operacji arytmetyczno-logicznych i jednocześnie miejscem przechowania wyniku, o ile kod rozkazu nie stanowi inaczej.

Mirosław Lach



Cd. ze str. 4

Zbigniew Świerzewski z Pudliszek pisze: Od niedawna jestem czytelnikiem miesięcznika "EdW". Długo szukałem odpowiedniego czasopisma, które potrafi jak najprościej przyswoić artykuły takim elektronikom jak ja (amatorom). To fajnie, że istnieje miesięcznik, który czyta się po prostu "od deski do deski" i z niecierpliwością czeka się na następny numer. Mam do redakcji "EdW" prośbę. Czy mógłbym uzyskać od Was (nawet odpłat-

nie) schemat regulatora temperatury: analogowego lub cyfrowego z wyświetlaczem LED lub LCD. (...)

Zbyszku, jesteśmy elektronikami nie od dziś i śmiemy przypuszczać, że sam schemat niewiele ci pomoże. Nie wiemy, do czego Ci ten regulator jest potrzebny, czym ma sterować, jaka ma być maksymalna temperatura. W serii modułów AVT opisywanych w Elektronice Praktycznej możesz znaleźć wszystkie "klocki" do budowy potrzebnego ci regulatora (seria AVT-104 i 147).

Możemy opracować w redakcji EdW dowolny regulator, wykonamy też płytki, zapewnimy części, i to nie tylko dla Ciebie, lecz i dla innych. Napiszcie więc, kochani jakiego regulatora się spodziewacie. Prosty regulator temperatury można wykonać na płytce wielofunkcyjnej PW-01, opisujemy go na stronie 9.

Nie wiemy, gdzie w kraju można kupić obudowy do wieży "Diora" - w Warszawie na Wolumenie piszący te słowa kupuje takowe u p. Czarka z Łodzi.

Cd. na str. 59

Adresy i adresowanie

Pojęcie adresu kojarzy się z listem, kopertą i listonoszem. Dokładnie adresujemy list, ponieważ nie znamy innego sposobu wyróżnienia odbiorcy, tak aby niezawodnie otrzymał on przeznaczony dlań wieści. Nie wystarczy napisać na kopercie: "Dla sympatycznego pana Henia" - trzeba podać nazwę miejscowości, ulicę, numer domu, ewentualny numer mieszkania.

Rodzajem adresu jest też numer telefonu, czyli *liczba*, którą trzeba wybrać za pomocą tarczy aparatu telefonicznego.

Co ważne, adres w postaci liczby jest zrozumiały dla mikroprocesora! Dla niego adres jest liczbą określającą jednoznacznie miejsce w pamięci - poszczególne komórki pamięci są więc ponumerowane. Pamięć ma zwykle organizację bajtową, to znaczy że pod jednym adresem zapisuje się lub odczytuje jednocześnie osiem bitów.

Adresowanie to dla nas, ludzi, napisanie adresu, czyli wskazanie odbiorcy. Podobnie należy rozumieć tę czynność wykonywaną przez mikroprocesor. Jednak sposoby adresowania są najprzeróżniejsze. Kilka z nich (nie wszystkie) przedstawimy poniżej. Na rysunkach linie przerywane dotyczą wystawiania adresu na szynę adresową, zaś linia ciągłą transmisję potrzebnych danych do akumulatora.

Niektóre ze sposobów adresowania mogą wydać się abstrakcyjne i niezrozumiałe, ale zapewniam - są przydatne. Rodzaje adresowań prześledzimy na przykładzie przesyłania danych do akumulatora.

Adresowanie bezpośrednie polega na zapisaniu w kodzie rozkazu adresu komórki pamięci danych (rys. 2).

Adresowanie pośrednie polega na zapisaniu w kodzie rozkazu adresu komórki pamięci, w którym znajduje się już właściwy adres (rys. 3).

Adresowanie rejestrowe jest odmianą adresowania pośredniego i polega na zawarciu w kodzie rozkazu umownego adresu (lub nazwy) rejestru.

Adresowanie indeksowe jest realizowane poprzez dodanie do adresu znajdującego się w kodzie rozkazu zawartości pewnej, wyróżnionej komórki pamięci, tzw. rejestru indeksowego (rys. 4).

Adresowanie natychmiastowe polega na zapisaniu stałej w kodzie rozkazu jako jednego z argumentów operacji, np. dodania do akumulatora liczby 5 (rys. 5).

Na rysunku 6 pokazano przykładową zawartość pamięci programu (zapisanej w EPROMIE) i pamięci danych (w pamięci RAM).

Należy zwrócić uwagę, że komórki pamięci programu mogą mieć te same numery adresów, co komórki pamięci danych, a przecież fizycznie są to dwie różne pamięci. Dekoder rozkazów "wie", o którą pamięć chodzi w danym rozkazie. Jednak nie zawsze pamiętają o tym początkujący programiści, co może powodować wiele zamieszania przy analizie działania programu. Dotyczy to też interpretacji rysunków 2...5.

W informatyce dane i adresy zapisuje się zwykle w postaci liczby szesnastkowej. Cyfry liczby szesnastkowej to dziesięć znanych nam cyfr systemu dziesiętnego oraz litery od A do F, które reprezentują liczby dziesiętne od 10 do 15. Wagi poszczególnych pozycji liczby szesnastkowej to $16^0=1$, $16^1=16$, $16^2=256$ itd. W celu łatwiejszego wyróżnienia tego zapisu,

liczby zaczynające się na literę poprzedza się cyfrą 0.

[okienko11]

Wspominając o kodzie szesnastkowym musimy powiedzieć o różnych kodach, które spotykamy w technice cyfrowej. Kody liczbowe to sposób zapisu informacji liczbowej. Jeśli informacja przetwarzana zawiera symbole literowe i znaki specjalne to taki kod nazwiemy kodem alfanumerycznym. Inne kody, które służą do wykrywania błędów i ewentualnego ich usuwania z informacji przesyłanej na pewną odległość (np. łączem telefonicznym) nazywają się kodami korekcyjnymi.

Możemy więc powiedzieć, że kody, jakie przyjdzie nam spotykać w praktyce mikroprocesorowej są zapisem informacji według pewnego, ściśle określonego przepisu.

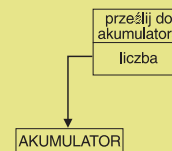
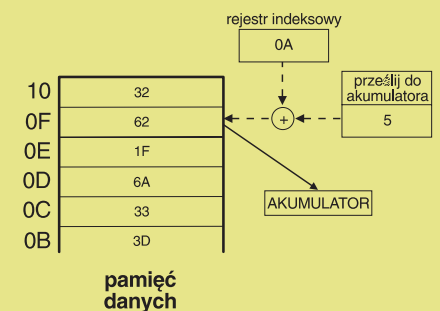
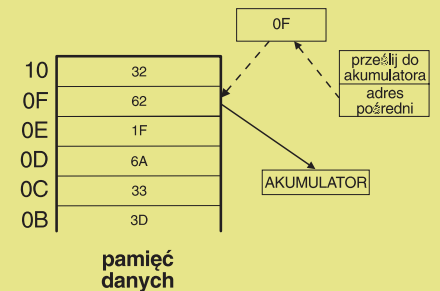
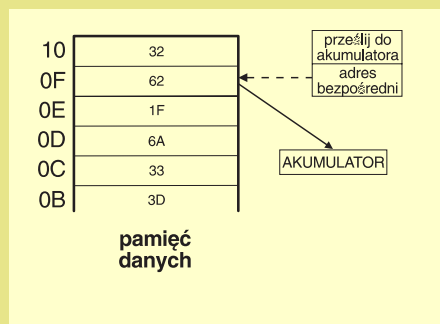
Najbardziej rozpowszechnionymi kodami liczbowymi są kody naturalne. Znanym wszystkim od przedszkola jest kod dziesiętny. Cały świat powszechnie go używa. Nie ma w nim nic ciekawego, ot, dziesięć cyfr i już. Jednak ten kod ma pewne cechy, które są wspólne dla wszystkich naturalnych kodów liczbowych. Po pierwsze, liczba zapisana w kodzie naturalnym jest ciągiem cyfr, czyli pozycji, z którego każda cyfra reprezentuje wielokrotność tzw. wagi danej pozycji. W kodzie dziesiętnym waga jest zawsze potęgą liczby 10, czyli patrząc od prawej strony liczby jest $10^0=1$, potem $10^1=10$, następnie $10^2=100$ itd. Nikt z nas jednak nie zastanawia się nad tym.

Każdy kod naturalny ma swoją podstawę. Podstawą jest podstawa potęg wagi. W kodzie dziesiętnym podstawa wynosi 10.

[koniec okienka 11]

Powiedzieliśmy sobie, że rozkazy są pobierane z pamięci, zatem przyszła kolej na omówienie pamięci. Zrobimy to za miesiąc.

Miroslaw Lach



10	PRZESUŃ W LEWO	10	32
0F	PRZESUŃ W LEWO	0F	62
0E	-4	0E	1F
0D	SKOCZ WZGLĘDNIE, JEŚLI CY=0	0D	6A
0C	3	0C	33
0B	ODEJMIJ OD AKUMULATORA	0B	3D
0A	34	0A	3F
9	ZERUJ BIT	9	67
8	45	8	0A5
7	DODAJ DO AKUMULATORA	7	0B4
6	11	6	11
5	10	5	1C
4	PORÓWNAJ	4	80
3	20	3	20
2	PRZEŚLIJ DO AKUMULATORA	2	30
1	06	1	0C
0	SKOCZ	0	82

pamięć programu

pamięć danych