IKS

INFORMATYKA KOMPUTERY SYSTEMY

CENA 100 zł

DODATEK „ŻOŁNIERZA WOLNOŚCI" NR 4/1988 ISSN 0860-2794

Specjalne programy graficzne umożliwiają tworzenie dźwięków na ekranie monitora, wygodną edycję parametrów itp. Komputer przetwarza pomysły muzyka na odpowiedni ciąg danych i przesyła do syntezatora. W tym miejscu powstaje bardzo ważny problem: w jaki sposób komputer ma nawiązać kontakt z syntezatorem? Odpowiedź w numerze: Nuty z komputera — str. 16

Liczby i tabele na wykresach — czternasty odcinek „W szponach Atari" tym razem poświęcony grafice, a w nim także 11 krótkich programów — str. 5

UWAGA! Kolejna edycja „Ligi Myślących" w tym numerze wyjątkowo na stronie 25.

Informatyczny krajobraz naszego kraju mieni się wszystkimi barwami. Ta dziedzina, jak żadna inna tętni dziś życiem, a to głównie dlatego, że szybko powstające, na ogół prężnie działające w warunkach rzeczywistej konkurencji firmy prześcigają się w oferowaniu usług i technicznych nowości. Komputery stały się interesem, i to zarówno dla tych, którzy nimi handlują, jak i tych, co postanowili zrobić z nich użytek. Stworzył się zdrowy rynek, na którym decydujący głos należy wreszcie do klienta.

Dziś o nabywcę trzeba zabiegać. Jesteśmy jedynym krajem socjalistycznym, w którym organizuje się tak wiele imprez handlowych. W Warszawie mieliśmy trzecią już wystawę komputerów. Początkowo „Agpol" zorganizował ją w salach hotelu Victoria, ale już po paru dniach jej trwania („Komputer '86") stało się oczywiste, że miejsca jest zbyt mało i to zarówno dla wystawców, jak i dla zainteresowanych. Kolejne wystawy odbywały się już w Pałacu Kultury i Nauki. Zainteresowanie nie słabnie, natomiast już bez głębszych analiz można zauważyć, że z roku na rok przybywa na niej profesjonalistów. Ci, którzy chcą sprzedać przyznają, że miły uśmiech i elegancka powierzchowność już nie wystarczają, choć korzystne kontrakty pomagają często załatwić zawiłości naszego prawa. Nietrudno się domyśleć, kto na tym traci. Na szczęście firmy funkcjonujące tylko na zasadzie pośrednictwa, łaskawie przystawiające pieczęci pozwalające załatwić sprawę w majestacie prawa, spotkać można coraz rzadziej. Zdecydowana większość, to firmy pragnące rzetelnie załatwić klienta — bo przecież z niego żyją.

Tłok panował również na „Infosystemie '88" — międzynarodowych targach elektroniki, telekomunikacji i techniki komputerowej, które po benefisie we wrocławskiej Hali Ludowej przeniosły się na tereny Międzynarodowych Targów Poznańskich.

Niestety amatorzy i miłośnicy informatyki niewiele z tych imprez skorzystają. Ceny nowoczesnego sprzętu wielokrotnie przerastają możliwości normalnej kieszeni. Jedynym pocieszeniem niech będzie fakt, że Spectrum czy Atari nie zdrożało (!) — co praktycznie równa się jego relatywnie niższej dziś cenie. Naszym Czytelnikom pozostaje Centralna Składnica Harcerska, giełda, sklepy Pewexu i nadzieja, że może kiedyś doczekamy się polskiego mikrokomputera. Czego i sobie życzymy.

SPIS TREŚCI

Temperatura, głos i komputery — str. 3
Dyskowy system operacyjny dla ZX Spektrum — str. 4
Wszponach Atari (14) — str. 5
Program — Słownik — str. 9
Lilavati — cz. 4 — str. 10
Gra w ośmiu hetmanów — str. 12
Przygotowanie drukarki GEMINI—70Xi — str. 15
Nuty z komputera (1) — str. 16
Programowanie mikroprocesora 6502 — str. 18
Kto pomoże? — str. 25
Liga Myślących — str. 25
Ocalony — str. 26
Program — Dźwięki. Klawiatura — str. 27
Spis najciekawszych programów — str. 28
Enigma — str. 31

TYLKO O 20 zł

Uprzejmie informujemy Czytelników, że rosnące ceny papieru i usług poligraficznych zmusiły nas do zmiany ceny.
Od czwartego numeru IKS kosztować będzie 100 zł. Cena w prenumeracie pozostanie bez zmian. Przepraszamy.

NAUKA • TECHNIKA • NAUKA

Już w 1990 roku w pierwszy rejs morski ma wyruszyć eksperymentalny japoński statek o nazwie YAMATO i wyposażony w nowego typu agregat, w którym wykorzystane zostanie zjawisko nadprzewodnictwa. Będzie on wyposażony w dwa silniki umieszczone na dnie statku, gdzie pobierać one będą wodę morską, która następnie zostanie wyrzucona na zewnątrz przy pomocy silnego pola magnetycznego. Wytwarzane ono będzie przez dwa silne elektromagnesy zbudowane z materiałów nadprzewodzących. Dzięki temu silnik zużywa nie tylko mniej energii elektrycznej, ale też zwiększa się znacznie jego moc. Zdaniem autorów projektu, w przyszłości tego typu statki będą w stanie rozwijać prędkości nawet do 100 węzłów, czyli ponad 180 km/godz.

■ ■ ■

W klinice uniwersyteckiej w Hamburgu testowane jest nowe urządzenie diagnostyczne tzw. "cyfrowa radiografia". Aparatura, podobnie jak inne obecnie stosowane urządzenia do badania układu kostnego, wykorzystuje również promienie rentgenowskie. Badanie sterowane jest jednak przez komputer, co umożliwiło zmniejszenie dawki promieniowania od 50 do 90 procent. W celu powiększenia niektórych badanych elementów układu kostnego nie trzeba przeprowadzać dodatkowych zdjęć rentgenowskich, lecz wystarczy wywołać z pamięci komputera potrzebny obraz detalu. Cyfrowa radiografia - zdaniem specjalistów - jest szczególnie przydatna w badaniach diagnostycznych dzieci oraz tych osób, u których ze względu na leczenie musi ono być często powtarzane.

■ ■ ■

Pracownicy Politechniki w LULEA (północna Szwecja) zerwali ze stereotypem, według którego paliwo do silników samochodowych powinno się spalać w wysokich temperaturach. Oryginalny pojazd szwedzkich inżynierów rozpoczyna jazdę dopiero wówczas, gdy paliwo w jego zbiorniku zamarznie. Prototyp pojazdu napędzanego lodem zaopatrzono w 25-litrowy zbiornik na wodę. W momencie zamarzania zwiększa ona swoją objętość, co powoduje wzrost ciśnienia w stalowej butli. Hydrauliczny mechanizm wykorzystuje podwyższone ciśnienie do uruchomienia napędu samochodu. Podczas prób niewielki samochodzik rozwijał prędkość do 50 km/godz., ale jego twórcy utrzymują, że przy drobnych zmianach konstrukcyjnych ten sam model powinien przekroczyć granicę 100 km/godz.

■ ■ ■

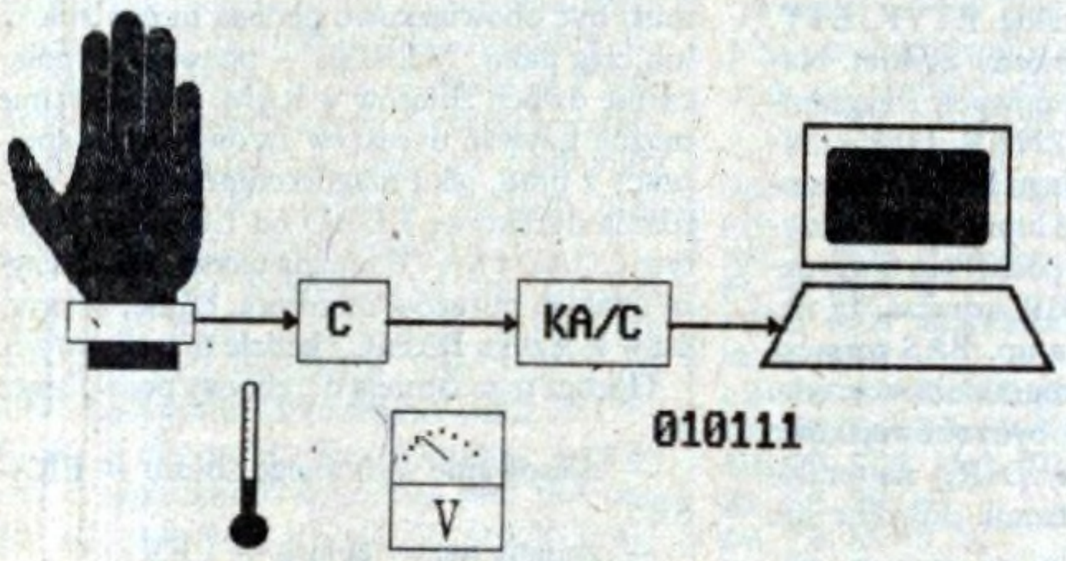
W zachodnioniemieckiej firmie SIEMENS opracowano nowy kompilator języka ADA. ADA jest językiem programowania powstałym na zamówienie amerykańskiego ministerstwa obrony między innymi w celu ograniczenia kosztów budowy i konserwacji oprogramowania wojskowych systemów informatycznych. Nowy kompilator charakteryzuje się budową modułową i może być wykorzystywany zarówno pod specjalizowanym systemem operacyjnym BS 2000, jak też przenoszony na inne systemy operacyjne i komputery.

TEMPERATURA, GŁOS I KOMPUTERY

Dziewięć, a może dziesięć lat temu w każdym razie na długo przed mikrokomputerami, w firmie mojej powstały warunki do realizacji „ambitnych planów młodego inżyniera informatyka". Wiązały się one z budową urządzeń, które umożliwiają wprowadzenie danych bezpośrednio do komputera, bez stosowania klawiatury lub innych papierowych nośników informacji. Innymi słowy chciałem, by informacje wprowadzane były do maszyny prosto ze źródła informacji, w moim przypadku od pacjenta, a ściślej z jego skóry. Przedmiotem pomiaru była bowiem temperatura osoby poddanej działaniu określonych czynników zewnętrznych, także farmakologicznych.

Dysponowałem wówczas odpowiednimi czujnikami temperatury. Zmieniały one jej wielkość na proporcjonalną wartość napięcia: od 0V do 5V. Gdy temperatura wynosiła 36°C czujnik „dawał" napięcie +0,5, gdy 43°C — 4,5V. Problem polegał zatem na tym, by uzyskane z czujników pomiary przekazać do komputera. Tradycyjnie odbywało się to w ten sposób, że odczytywane wielkości były notowane na papierze, by potem wprowadzić je do komputera za pośrednictwem klawiatury. Procedura ta była oczywiście bardzo pracochłonna i trudno było wykorzystać komputer do śledzenia temperatury w czasie rzeczywistym, na przykład w celu alarmowania, gdy wynik pomiaru przekroczył wartość 38.

W celu realizacji tego sprzężenia, czujnik — komputer, niezbędny okazał się konwerter A/C (analog/cyfra) — urządzenie elektroniczne, które „zmienia" napięcie na odpowiadającą jego wartości liczbę. Jest ona w takiej postaci (binarnej) jaką „akceptuje" każdy komputer.

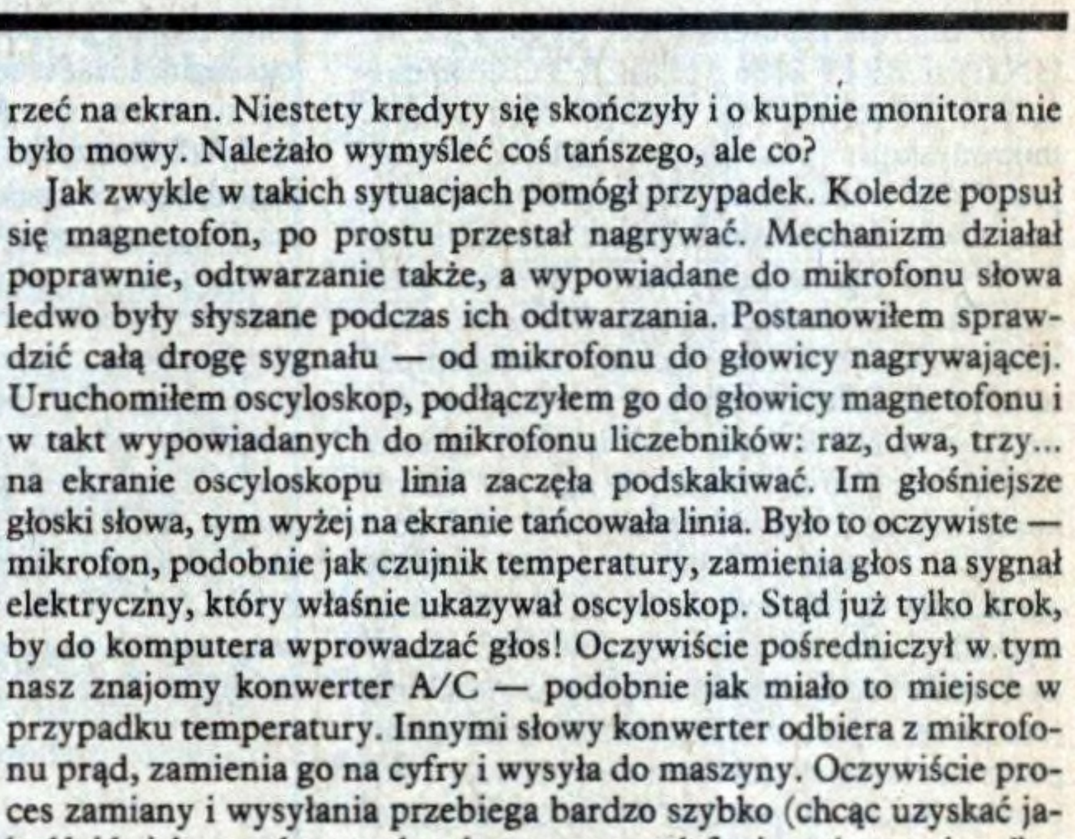


W praktycznej realizacji tego sprzężenia maszyna co pewien czas wysyła impuls „prośbę" do konwertera A/C, by „zechciał zobaczyć", jaka jest wartość temperatury. W odpowiedzi konwerter A/C zamienia napięcie (odpowiednie temperaturze) na liczbę i wysyła ją do komputera (za pośrednictwem interfejsu). Pomiar został dokonany — bez udziału człowieka, maszyna odczytała temperaturę i może ją dalej analizować.

Po podłączeniu czujnika do konwertera z jednej strony, komputera z drugiej strony, okazało się, że zestaw ten pracuje nadzwyczajnie — cicho, poprawnie i bardzo szybko. Poszedłem do pacjenta, który siedział w pokoju znacznie oddalonym od komputera, uśmiechnęliśmy się do siebie i... zwątpiłem czy na pewno komputer mierzy i rejestruje temperaturę. Szybko pobiegłem na salę maszyny, uruchomiłem drukarkę. Okazało się, że sprzężenie działa bez zarzutu — temperatura wolno się zmieniała. Wróciłem do pacjenta. Po chwili znowu byłem pełen wątpliwości, czy maszyna nadal odbiera informacje o temperaturze?

Oczywiście najprościej było by zainstalować obok pacjenta monitor sprzężony z komputerem i po prostu od czasu do czasu spojrzeć na ekran. Niestety kredyty się skończyły i o kupnie monitora nie było mowy. Należało wymyśleć coś tańszego, ale co?

Jak zwykle w takich sytuacjach pomógł przypadek. Koledze popsuł się magnetofon, po prostu przestał nagrywać. Mechanizm działał poprawnie, odtwarzanie także, a wypowiadane do mikrofonu słowa ledwo były słyszane podczas ich odtwarzania. Postanowiłem sprawdzić całą drogę sygnału — od mikrofonu do głowicy nagrywającej. Uruchomiłem oscyloskop, podłączyłem go do głowicy magnetofonu i w takt wypowiadanych do mikrofonu liczebników: raz, dwa, trzy... na ekranie oscyloskopu linia zaczęła podskakiwać. Im głośniejsze głoski słowa, tym wyżej na ekranie tańcowała linia. Było to oczywiste — mikrofon, podobnie jak czujnik temperatury, zamienia głos na sygnał elektryczny, który właśnie ukazywał oscyloskop. Stąd już tylko krok, by do komputera wprowadzać głos! Oczywiście pośredniczył w tym nasz znajomy konwerter A/C — podobnie jak miało to miejsce w przypadku temperatury. Innymi słowy konwerter odbiera z mikrofonu prąd, zamienia go na cyfry i wysyła do maszyny. Oczywiście proces zamiany i wysyłania przebiega bardzo szybko (chcąc uzyskać jakość dźwięku porównywalną do rozmowy telefonicznej, „zamiany" te powinny odbywać się z szybkością około 10 tys. w czasie 1 sek.).



To był pierwszy krok w konstruowaniu taniego urządzenia wyjściowego komputera, które pozwoliłoby mi na bieżące śledzenie, z dużej odległości, co komputer w danej chwili robi. Dysponując zatem sprzężeniem: mikrofon — konwerter A/C — komputer, z łatwością mogłem przekazać do pamięci maszyny wyrazy: jeden, dwa, trzy, ... dziewięć oraz trzydzieści i czterdzieści. W ten sposób maszyna „poznała" wzorce słownej interpretacji niektórych liczb. Wykorzystując odpowiedni program, komputer z łatwością mógł łączyć poszczególne słowa i generować cyfrowe odwzorowania słów: od trzydzieści do czterdzieści dziewięć.

Komputer dysponując wzorcami liczebników mógł je oczywiście odtwarzać — „dawać" na drutach interfejsu w postaci liczb binarnych. Ale przecież tego nie słychać! To są tylko pewne poziomy napięć — dla jedynek +5V, a dla zer 0V. Należało zatem wykonać zabieg odwrotny, niż to uczynił konwerter A/C. Cyfry podawane przez interfejs trzeba było z powrotem zamienić na sygnał elektryczny — taki jaki dawał wcześniej mikrofon. Okazało się to niezwykle proste — korzystając z konwertera C/A. Układ ten zamienia liczby binarne (odbierane z interfejsu komputera) na przebieg analogowy — ciągły sygnał elektryczny, taki jaki wcześniej uzyskiwałem z mikrofonu. Pozostało zatem tylko dodanie do konwertera głośnika i... komputer zaczął gadać i to jak! Szybko podłączyłem wyjście konwertera C/A do wzmacniacza, pociągnąłem przewód z sali komputerowej do pacjenta (ten ciągle na szczęście czekał), założyłem słuchawki. Słyszałem: trzydzieści sześć, trzydzieści sześć siedem, trzydzieści sześć sześć siedem, ciąg dalszy nastąpi...

PRAKTYK

Dyskowy system operacyjny dla ZX Spectrum

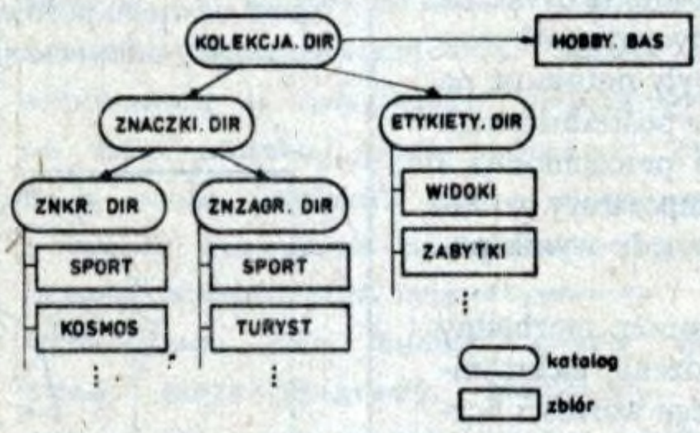
Do mikrokomputerów ZX SPECTRUM, UNIPOLBRIT 2086 i TIMEX TC 2048 może być dołączona tego samego typu pamięć mikrodyskowa (pmd) firmy TIMEX. Współpraca pmd z mikrokomputerem odbywa się na podstawie dyskowego systemu operacyjnego DOS (disk operating system). W skład pmd wchodzą: sterownik, napędy dyskietek, zasilacz i sprzęg (interface) łączący mikrokomputer z pmd. Całość może być w jednej obudowie lub stanowić oddzielne moduły łączone między sobą przewodami. Z jednym mikrokomputerem może współpracować od 1 do 4 napędów dyskietek. Poszczególne napędy oznaczone są literami A, B, C, D. W takim wypadku jeden zasilacz obsługuje dwa napędy dyskietek. Pracą napędów zarządza sterownik (controber). Jest to samodzielny mikrokomputer zawierający mikroprocesor 280A i 16kB pamięci RAM. W pamięci tej rezyduje system operacyjny DOS. Sterownik ma dodatkowo wyprowadzone na zewnątrz dwa łącza typu RS-232C, umożliwiające współpracę z innymi komputerami oraz urządzeniami zewnętrznymi takimi jak: modemy, drukarki, plotery itp.

Nośnikiem informacji jest dyskietka o średnicy 3 cali. Jest to elastyczny krążek z tworzywa sztucznego, pokryty materiałem magnetycznym i zamknięty w sztywnej plastikowej kasecie. W kasecie znajdują się dwa okienka, przez które dyskietka ma styczność z głowicami odczytu — zapisu odpowiedniego napędu. Gdy dyskietka znajduje się na zewnątrz napędu, to okienka są zamknięte zasłonami i chronią powierzchnię roboczą dyskietki przed zanieczyszczeniami. W czasie wsuwania dyskietki do napędu, zasłony automatycznie otwierają się i umożliwiają styk głowic z dyskietką.

Informacja na dyskietce jest zapisana na dwóch powierzchniach oznaczonych literami A i B. Napęd dyskietki umożliwia jednocześnie dostęp tylko do jednej powierzchni A lub B. Pojemność jednej powierzchni wynosi 160 kB, a całej dyskietki 320 kB. Powierzchnia robocza dyskietki podzielona jest przez DOS w trakcie jej formatowania na 40 ścieżek po 16 sektorów i po 256 bajtów w sektorze. W czasie odczytu lub zapisu dyskietka wiruje ze stałą prędkością. Dyskietka ma mechaniczne zabezpieczenia przed możliwością niepożądanego zapisu w postaci małych plastikowych przycisków. Wciśnięcie przycisku uniemożliwia zapis na dyskietce.

Producenci dyskietek zalecają, aby dyskietek nie poddawać działaniu pól magnetycznych, wysokiej temperaturze, nie dotykać żadnymi przedmiotami powierzchni roboczych i chronić przed kurzem i wilgocią. Przestrzeganie tych zaleceń zapewni niezawodność użytkowania dyskietki.

Informacje zapisywane na dyskietkach (programy, zbiory danych) są organizowane przez DOS w odpowiednie struktury danych. Dane można grupować w katalogach, te katalogi zagnieżdżać w innych katalogach itp. Tworzy się w ten sposób „drzewo struktury danych". Zaletą takiej organizacji jest możliwość dogodnego podziału istniejących grup informacji i programów wg nazw katalogów i zbiorów. Schemat takiej struktury przedstawia poniższy rysunek. Na rysunku przedstawiono przykładową strukturę katalogów i zbiorów, utworzoną dla potrzeb np. kolekcjonera znaczków pocztowych i etykietek zapałczanych. Katalogiem głównym jest tu katalog o nazwie KOLEKCJA. DIR, który zawiera dwa katalogi niższego poziomu o nazwach ZNACZKI. DIR i ETYKIETY. DIR oraz program obsługujący te zbiory o nazwie HOBBY. BAS. Z kolei katalog ZNACZKI.



DIR znów zawiera dwa katalogi ZNKR. DIR I ZNZAGR. DIR, a katalog ETYKIETY. DIR zawiera już nazwy zbiorów etykiet. Nazwy zbiorów znaczków krajowych i zagranicznych zawierają katalogi ZNKR. DIR i ZNZAGR. DIR. Nazwa katalogu lub zbioru może zawierać maksymalnie 8 liter i cyfr i zaczyna się od litery. Po kropce podaje się tzw. rozszerzenie. Rozszerzenie DIR oznacza, że nazwa jest nazwą katalogu, a np. BAS oznacza nazwę zbioru będącego programem w języku BASIC. Katalogi muszą być obowiązkowo oznaczone rozszerzeniem DIR, natomiast zbiory mogą mieć rozszerzenie dowolne lub nie posiadać ich wcale.

Dojście do zbioru znajdującego się na najniższym poziomie odbywa się poprzez określenie tzw. drogi do zbioru lub ciągu nazw katalogów i zbioru. Przykładowo, aby dojść do zbioru znaczków KOSMOS, należy określić ciąg nazw w sposób następujący: KOLEKCJA. DIR: ZNACZKI. DIR: ZNKR. DIR: KOSMOS

Tworzenie katalogów i zbiorów odbywa się za pomocą dyrektywy DIM i podania ciągu nazw. Na przykład, aby utworzyć katalog ZNKR. DIR, trzeba przedtem utworzyć katalogi KOLEKCJA. DIR I ZNACZKI. DIR. Będzie to wyglądało następująco:
DIM „KOLEKCJA.DIR" — utworzenie katalogu KOLEKCJA.DIR
DIM „KOLEKCJA.DIR: ZNACZKI.DIR" — utworzenie katalogu ZNACZKI.DIR
DIM „KOLEKCJA.DIR: ZNACZKI.DIR: ZNKR.DIR" — utworzenie katalogu ZNKR.DIR

Informację o zawartości katalogów uzyskujemy za pomocą dyrektywy CAT i odpowiedniego ciągu nazw do katalogu. Chcąc uzyskać informację o zawartości katalogu ZNZAGR.DIR polecenie należy sformułować następująco:
CAT „KOLEKCJA.DIR: ZNACZKI.-DIR: ZNZGR.DIR"

W wyniku wykonywania tej dyrektywy na ekranie wyświetli się wykaz zbiorów znajdujących się w katalogu ZNZAGR.DIR z informacją o wielkości zbioru w bajtach i o wielkości pamięci zajętej przez ten zbiór. Występuje tu pewna rozrzutność w gospodarowaniu pojemnością dyskietki przez DOS, który na zbiór rezerwuje obszar w pełnych kB i np. zbiór o wielkości 50 B będzie miał zarezerwowane 1 kB, a zbiór o 1040B — 2 kB.

Zbiory mogą być programowo zabezpieczone przed zapisem lub odczytem. Ochrona przed odczytem polega na tym, że nazwy zbiorów nie wpisują się w trakcie działania dyrektywy CAT, są niewidoczne. Ochrony zbiorów przed zapisem lub odczytem i zdjęcie tej ochrony dokonuje się za pomocą dyrektywy ATTR.

Zabezpieczenie programowe nie chroni przed zniszczeniem informacji dyrektywą FORMAT. Dyrektywa ta przygotowuje dyskietkę do pracy z DOS. Wyznacza ona ścieżki i sektory dyskietki oraz zeruje dyskietkę. Czynność ta nazywa się formatowaniem dyskietki. Ponieważ dyrektywa ta niszczy całą zawartość dyskietki, należy się dobrze zastanowić przed jej formatowaniem, aby nie zniszczyć przypadkowo ważnych informacji.

Zapisu do zbioru, odczytu ze zbioru, łączenia dwóch zbiorów dokonuje się za pomocą dyrektyw podobnych jak przy współpracy z magnetofonem kasetowym. I tak SAVE — powoduje zapis do zbioru, LOAD — powoduje przepisanie zbioru do pamięci LOAD musi być obowiązkowo podana nazwa zbioru lub ciąg nazw. MERGE — pozwala na połączenie dwóch zbiorów w RAM. W programie można używać dyrektyw zarówno do współpracy z pmd, jak i magnetofonem. DOS rozróżnia dyrektywę LOAD od LOAD. Dyrektyw LOAD i SAVE można używać do odczytu i zapisu obrazów monitora, tablic, programów w języku BASIC i kodzie maszynowym.

Oprócz tego istnieją dyrektywy pozwalające na:
— skasowanie wybranego zbioru — ERASE
— zmiany nazwy zbioru — LET
— kopiowanie zbiorów z jednego katalogu do innego — MOWE

Dla ułatwienia pracy można tworzyć program startu, który automatycznie uruchamia się po wczytaniu i naciśnięciu przycisku RESET na sprzęgu pmd. Program ten może ściągać z dyskietki do RAM potrzebne do dalszej pracy zbiory i programy, wyświetlać listę czynności, itp.

Jak więc widać, pmd ma wiele zalet, zapisywanie i wczytywanie programów z dyskietki jest wielokrotnie szybsze w porównaniu z magnetofonem i bardziej niezawodne. Programy mogą się w każdej chwili komunikować z dyskiem i pobierać potrzebne dane, a także zapisywać wyniki obliczeń. Wadą zaś jest zbyt wysoka cena. Pamięć mikrodyskowa jest kilkakrotnie droższa od samego mikrokomputera.

J.R.

W szponach Atari (14)

Graficzna prezentacja danych liczbowych

Tomasz MROWIEC, Ludwik PIELA

Dane, przedstawione w postaci kolumn liczb lub tablicy, trudniej jest zwykle analizować niż ich graficzną prezentację w postaci wykresów. Te ostatnie pozwalają szybciej ocenić informację zawartą w zbiorze liczb. Umożliwiają poglądowe przedstawienie różnorodnych zależności między danymi, które często trudno wykryć w kolumnie wydrukowanych liczb.

Programy przeznaczone do graficznej prezentacji danych liczbowych wchodzą w skład oprogramowania większości komputerów. O ich przydatności najlepiej świadczy fakt, że są elementem składowym większości pakietów zintegrowanych „poważnych" mikrokomputerów. Również komputery domowe, tej klasy co Atari serii XL lub XE, posiadają zwykle kilka programów tego typu. Dla Atari są to, na przykład: GRAPH IT, SYNGRAPH oraz B/GRAPH. Zwykle są to programy uniwersalne, przeznaczone albo wyłącznie do graficznej prezentacji danych (GRAPH IT), albo do obliczeń statystycznych z możliwością wyprowadzania wyników w postaci graficznej (B/GRAPH).

Często zdarza się jednak, że chcemy przedstawić w postaci graficznej wyniki obliczeń własnego programu. Wtedy musimy włączyć do niego moduł tworzący odpowiednie wykresy, albo dysponować programem, który przyjmie wyniki w charakterze danych wejściowych i przedstawi je w postaci graficznej. Dlatego spróbujemy zaprezentować kilka metod i przykładowych programów umożliwiających uzyskanie kilku podstawowych typów wykresów i diagramów, od najprostszych do dosyć wyrafinowanych. Mamy nadzieję, że czytelnicy zaadaptują je do własnych potrzeb oraz uzupełnią o dodatkowe możliwości.

Do tworzenia wykresów można używać dowolnego trybu graficznego komputera Atari oraz odpowiednich instrukcji, bądź PRINT do trybów tekstowych, bądź PLOT i DRAWTO dla graficznych. Chociaż w pewnych przypadkach wygodnie jest posługiwać się instrukcją PRINT, to środki graficzne pozwalają tworzyć wykresy bardziej szczegółowe.

Do najprostszych należą wykresy prezentujące ogólne tendencje lub charakter zmian danych, zwykle bez żadnego lub z minimalnym opisem. Mogą one przedstawiać, na przykład, obraz zmian jakiejś wielkości w zadanym okresie. Można go tak umieścić, aby wielkości odkładały się albo w poziomie, albo w pionie. W pierwszym przypadku okresy czasu odliczane są od góry do dołu, a odpowiadające im wielkości — w kierunku poziomym od lewej do prawej.

Przybliżony wykres wartości można utworzyć wyświetlając wybrane symbole w miejscach ekranu odpowiadających ich względnym wartościom. Załóżmy, że postanowiliśmy umieszczać wielkości poziomo na ekranie, który ma 24 wiersze po 40 znaków. Pozycje symboli w linii numerujemy od lewej do prawej od 0 do 39, a wiersze — z góry na dół od 0 do 23. Będziemy wykreślać tylko w wierszach nieparzystych, a w nich dla przedstawienia wartości — pozycje od 8 do 38. W tym przypadku maksymalna wielkość odpowiadać będzie 38 pozycji, a minimalna — 8 pozycji.

W celu zobrazowania wielkości w zakresie od 8 do 38 pozycji musimy wykonać skalowanie. Możemy to zrobić według następującego wzoru:

N=(W—Wm)*Zp/Zw+Nm

gdzie:

N — numer pozycji w wierszu,
W — kreślona wielkość,
Wm — wielkość minimalna,
Zp — zakres numerów pozycji,
Zw — zakres wielkości,
Nm — pozycja minimalna.

W naszym przykładzie zakres wartości przedstawianych danych równy jest WM — Wm, zakres numerów pozycji 38—8=30 i numer pozycji minimalnej — 8. A zatem numer pozycji wydruku symbolu gwiazdki („*") w wierszu, dla dowolnej wartości przedstawianych danych, można wyznaczyć według wzoru:

Np=(W—Wm)•30/(WM—Wm)+8

```
FK 1 REM ****************
FJ 2 REM *              *
XA 3 REM * Program nr 1 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
LS 40 GRAPHICS 0
MA 50 R=(38-8)/(651-99)
GW 60 FOR K=1 TO 10
XQ 65 PRINT
DH 70 READ S
MX 80 P=INT((S-99)*R+8+0.5)
EM 90 FOR I=1 TO P-1:PRINT " ";:N
   EXT I
EE 100 PRINT "*"
GP 120 NEXT K
WU 130 DATA 210,150,99,250,183,35
   2,410,390,300,651
NG 140 GOTO 140
```

Program nr 1 tworzy przybliżony wykres wykorzystując powyższy wzór. Dla każdej wartości wyznaczana jest pozycja wydruku symbolu gwiazdki, która zaokrąglona jest do najbliższej wartości całkowitej za pomocą funkcji INT. Po pewnych modyfikacjach program może pracować z dowolnym zbiorem danych, przy dowolnych wymiarach ekranu. W tym celu w podanym wzorze należy wykorzystać wartości minimalną i maksymalną oraz numery pozycji wprowadzone instrukcją INPUT.

Aby otrzymać wykres, w którym wartości odkładają się w pionie, wprowadzimy do powyższego programu dwie zmiany. Po pierwsze, dla każdego elementu danych określimy odpowiedni numer wiersza. Po drugie, w tej linii wybierzemy numer pozycji, odpowiadający zadanemu okresowi czasu. W ten sposób każdy element danych będzie przedstawiony na ekranie symbolem gwiazdki, w pozycji o obliczonych numerach wiersza i kolumny.

Do wskazania odpowiedniej pozycji gwiazdki na ekranie można wykorzystać instrukcję POSITION, która określa położenie symbolu na ekranie dla następnej instrukcji PRINT.

Dla danych, używanych w programie nr 1 i dla ekranu zawierającego 24 wiersze po 40 znaków, wybierzemy dla miesięcy każdą trzecią kolumnę, zaczynając od 3, a kończąc na 36. Na wykres wykorzystamy 20 górnych linii ekranu. W tym przypadku wartości maksymalnej (724) odpowiadać będzie pozycja 33 najwyższego wiersza ekranu (o numerze 1). Numery pozycji wzdłuż każdego wiersza określa się jako 3•M, gdzie M — numer miesiąca. Dla skalowania wartości w zakresie pierwszych 20 wierszy ekranu wykorzystamy następujący wzór:

Nw=(WM—W)•Zw/Z+Sm

gdzie

Nw — numer wiersza,
WM — wartość maksymalna
W — wartość bieżąca,
Zw — zakres wierszy,
Z — zakres wartości,
Sm — minimalny numer wiersza.

```
FK 1 REM ****************
FJ 2 REM *              *
XU 3 REM * Program nr 2 *
FL 4 REM *              *
FO 5 REM ****************
NL 6 REM
CL 50 GRAPHICS 0:POKE 752,2
HP 60 T=(20-1)/(724-99)
IT 70 FOR M=1 TO 12
DI 80 READ S
MT 90 WI=INT((724-S)*T+1+0.5)
SZ 100 KO=M*3
PP 110 POSITION KO,WI
KC 120 PRINT "*";
HL 130 NEXT M
LY 140 DATA 210,150,99,250,183,35
   2,410,390,300,651,724,516
NT 150 GOTO 150
```

W tym przykładzie zakres numerów wierszy wynosi 19 (czyli 20—1), minimalny numer wiersza równy jest 1. Opisane zadanie realizuje program nr 2. Można go tak zmodyfikować, aby zapewnić dowolne położenie wykresu. W takim przypadku żądane liczby wierszy i kolumn wydruku wprowadza się za pomocą instrukcji INPUT. Zakres wartości danych i wartość maksymalna mogą być określone przez program podczas wprowadzania danych.

Powyższe przykłady byłyby bardzo przydatne, gdybyśmy chcieli pracować wyłącznie w trybie tekstowym. Możemy jednak wybrać jeden z wielu trybów graficznych i wykorzystać specjalne instrukcje graficzne w celu działania na współrzędnych punktów elementarnych, zamiast na numerach wierszy i kolumn. Zyskamy wtedy na dokładności wykresu. Spróbujmy zatem utworzyć wykres, podobny do powyższych, wykorzystując instrukcję PLOT. Załóżmy, że rozdzielczość ekranu równa jest 160 punktów w poziomie i 80 punktów w pionie (GRAPHICS 7). Wykres rozmieścimy tak.

aby wykorzystać linie od 9 do 79 i kolumny od 10 do 120. Dla oznaczenia miesięcy użyjemy co 10 punktu zaczynając od 10. W celu wyskalowania wartości danych w zakresie od 0 do 79 wykorzystamy następujący wzór:

Y=(WM—W)*Zw/Z+Ym

Omawiany, bardzo uproszczony, wykres pionowy kreślony jest przez program nr 3. Aby otrzymać wykres, w którym wartości danych odkładają się w poziomie, należy zamienić miejscami współrzędne X i Y, biorąc pod uwagę rozmiary ekranu.

```
FK 1 REM ****************
FJ 2 REM *                *
YO 3 REM * Program nr 3 *
FL 4 REM *                *
FO 5 REM ****************
NL 6 REM
YG 50 GRAPHICS 7:COLOR 1
HL 60 R=(40-0)/(724-99)
HA 70 X=10
IE 80 FOR K=1 TO 12
DJ 90 READ S
VA 100 Y=INT((724-S)*R+0.5)
SU 110 PLOT X,Y
PO 120 X=X+10
GR 130 NEXT K
LY 140 DATA 210,150,99,250,183,35
   2,410,390,300,651,724,516
OA 150 END
```

Punkty wykresu można połączyć między sobą odcinkami prostych. Taki wykres, zwany liniowym, tworzony jest przez program nr 4.

```
FK 1 REM ****************
FJ 2 REM *                *
ZI 3 REM * Program nr 4 *
FL 4 REM *                *
FO 5 REM ****************
NL 6 REM
YG 50 GRAPHICS 7:COLOR 1
GL 60 DIM X(12),Y(12)
HM 70 R=(40-0)/(724-99)
WV 80 X1=20
IV 90 FOR M=1 TO 12
ZE 100 READ S
DQ 110 Y(M)=INT((724-S)*R+0.5)
VK 120 X(M)=X1
IK 130 X1=X1+10
HN 140 NEXT M
XN 150 PLOT X(1),Y(1)
TI 160 FOR K=2 TO 12
VV 170 DRAWTO X(K),Y(K)
HB 180 NEXT K
MI 190 DATA 210,150,99,250,183,35
   2,410,390,300,651,724,516
NR 200 END
```

Najprostsze metody, opisane powyżej, przydatne są do szybkiego tworzenia prostych wykresów i przedstawiania charakteru zmian danych. Zawierają one jednak zbyt mało informacji ilościowej. Zwykle interesuje nas uzyskanie z wykresu informacji bardziej szczegółowych. Wyskalowanie i opisanie osi współrzędnych pozwala dokładniej określić zależności między wielkościami, a także dokonać interpolacji danych.

Opisywanie wykresów wymaga pewnych zmian w równaniach skalowania. Wybierając zakres wartości należy uwzględnić opis osi współrzędnych. Jeśli, na przykład, przetwarzamy zbiór liczb o zakresie wartości (—96, 89), a osie współrzędnych opisane są w przedziale (—100, 100), to w równaniach należy podstawić zakres wartości równy 200. Analogicznie, wartość minimalna będzie równa —100, a maksymalna — 100.

Osie współrzędnych możemy wykreślić za pomocą ciągu symboli „+", lub „—", tak jak w programie nr 5. Kreśli on wykres punktowy w trybie tekstowym, pomimo tego osie są wyskalowane i opisane.

```
FK 1 REM ****************
FJ 2 REM *                *
YO 3 REM * Program nr 5 *
FL 4 REM *                *
FO 5 REM ****************
NL 6 REM
FR 40 DIM M$(3),CLS$(1):CLS$=CHR$
   (125):POKE 752,1
VO 45 ? CLS$
YJ 50 POSITION 11,0:? "ROCZNY WYK
   RES CEN"
CD 70 POSITION 6,2:? "    1   2
    3   4   5   6   7   8"
VR 80 POSITION 6,3:? "    0   0
    0   0   0   0   0   0"
IP 90 POSITION 6,4:? "0   0   0
    0   0   0   0   0   0"
NM 100 POSITION 6,5:? "+---+---+-
   --+---+---+---+---+---+"
WY 110 R=(38-6)/(800-0)
HN 120 MIN=10000
BE 130 MAX=0
HN 140 T=0
SV 150 FOR K=1 TO 12
WJ 160 READ M$,S
PB 170 IF S<MIN THEN MIN=S
SZ 180 IF S>MAX THEN MAX=S
HO 190 T=T+S
HC 200 P=INT((S-0)*R+6+0.5)
DD 210 POSITION 0,K+5:? M$;"   :"
   ;:POSITION P,K+5:? "*";:POSITI
   ON 38,K+5:? ":";
GQ 220 NEXT K
IV 230 POSITION 6,18:? "+---+---+
   ---+---+---+---+---+---+"
NT 250 POSITION 11,20:? "CENA MIN
   IMALNA ";:POSITION 29+(3-LEN(S
   TR$(MIN))),20:? MIN;
UH 260 POSITION 11,21:? "CENA MAK
   SYMALNA ";:POSITION 29+(3-LEN(
   STR$(MAX))),21:? MAX;
KX 270 POSITION 11,23:? "CENA SRE
   DNIA ";:POSITION 29+(3-LEN(STR
   $(INT(T/12)))),23:? INT(10*T/1
   2)/10;
PR 280 GOTO 280
FB 290 DATA STY,210,LUT,155,MAR,7
   9,KWI,150,MAJ,175,CZE,252,LIP,
   510,SIE,280,WRZ,300,PAZ,751,LI
   S,724,GRU,726
```

Wygodniejszym sposobem kreślenia osi współrzędnych jest użycie trybu graficznego o dużej rozdzielczości. Pojawia się jednak pewien problem z opisywaniem wykresów. W komputerach Atari za pomocą instrukcji języka BASIC nie możemy, w sposób bezpośredni, umieścić znaków alfabetycznych w dowolnym miejscu ekranu (za wyjątkiem okna tekstowego). Musimy do tego wykorzystać specjalny podprogram „kreślący" te znaki. W programie nr 6, i dalszych, zaczyna się on od linii 10000. Przed wywołaniem go należy określić współrzędne położenia początku tekstu (X i Y w liniach ekranu), a drukowany tekst umieścić w zmiennej tekstowej T. Niestety, takie rozwiązanie ogranicza możliwości używania różnych trybów graficznych. Wykresy z opisami (czytelnymi) możemy uzyskać tylko w trybie graficznym 8.

```
FK 1 REM ****************
FJ 2 REM *                *
AW 3 REM * Program nr 6 *
FL 4 REM *                *
FO 5 REM ****************
NL 6 REM
AD 60 DIM X(12),Y(12),T$(40),TX$(
   21)
NT 80 GRAPHICS 8+16:COLOR 1
CW 100 T$="WYKRES PRODUKCJI":X=15
   :Y=1:GOSUB 10000
EH 110 PLOT 80,27:DRAWTO 80,155
WE 120 PLOT 304,155:DRAWTO 304,27
QF 140 FOR Y=27 TO 155 STEP 16
WF 150 PLOT 79,Y:DRAWTO 304,Y
MH 160 NEXT Y
RA 170 REM
AP 180 Y=19
WL 190 FOR S=0 TO 800 STEP 100
XE 200 T$="    ":T$(4-LEN(STR$(S))
   )=STR$(S)
AP 210 X=6:GOSUB 10000
YO 220 Y=Y-2
JU 230 NEXT S
QV 240 REM
OA 250 FOR CO=12 TO 34 STEP 2
BZ 260 X=CO:Y=19:T$="+"
SP 270 GOSUB 10000
LL 280 NEXT CO
NJ 290 T$="S L M K M C L S W P L
   G":X=12:Y=20:GOSUB 10000
GM 300 T$="T U A W A Z I I R A I
   R":X=12:Y=21:GOSUB 10000
YY 310 T$="Y T R I J E P E Z Z S
   U":X=12:Y=22:GOSUB 10000
IY 320 TX$="WARTOSC PRODUKCJI"
SI 330 FOR K=1 TO 17:T$=TX$(K,K)
BM 340 X=2:Y=K+2:GOSUB 10000
GX 350 NEXT K
ZX 370 T=(155-27)/(800-0)
TQ 380 X1=96
TF 390 FOR K=1 TO 12
UV 400 X(K)=X1
LD 410 X1=X1+16
ZL 420 READ S
JN 430 Y(K)=INT((800-S)*T+27+0.5)
GW 440 NEXT K
SH 450 FOR K=1 TO 11
BY 460 PLOT X(K),Y(K):DRAWTO X(K+
   1),Y(K+1)
MG 480 PLOT X(K),Y(K)-1:DRAWTO X(
   K+1),Y(K+1)-1
HG 490 NEXT K
LU 500 DATA 210,150,99,250,183,35
   2,410,390,300,651,724,516
NL 510 GOTO 510
WV 9999 REM ZNAKI W GRAFICE 8
ZT 10000 START=PEEK(89)*256+PEEK(
   88)+40*Y*8+X
```

Podczas tworzenia wykresów z opisem osi należy przestrzegać pewnych reguł. Opis powinien być prosty i krótki. Większa ilość objaśnień komplikuje wykres i utrudnia jego analizę. Tekst wyjaśniający należy umieszczać, w miarę możliwości, w tym wierszu lub obszarze co obrazowane dane, a nie wynosić go w dodatkowe tablice lub opisy. Krok opisu osi współrzędnych należy wybrać dogodnym do analizy wykresu, na przykład, krok opisu równy 10 będzie wygodniejszy, niż krok opisu równy 7. Analizę ułatwia zaznaczenie początku układu współrzędnych. Przydatne jest również, szczególnie dla ułatwienia interpolacji danych, naniesienie podziałów na osiach współrzędnych. Linie wykresu powinny być grubsze lub jaśniejsze niż osie lub linie siatki współrzędnych.

Powyższe zalecenia zostały uwzględnione podczas tworzenia wykresu programem nr 6.

Wygodnym sposobem, ułatwiającym analizę, jest przedstawienie danych w postaci zbioru kolumn, zamiast oddzielnych punktów. Sposób ten ilustruje program nr 7, wykorzystujący instrukcję PRINT do tworzenia wykresu.

Przykładem tworzenia histogramu (wykresu słupkowego) w trybie graficznym jest program nr 8. W otrzymanym wykresie szerokość kolumn równa jest szerokości odstępu między nimi. Należy trzymać się tej zasady, ponieważ wąskie słupki są mniej poglądowe.

Wygląd lub poglądowość kreślonych histogramów można poprawić za pomocą kolorów. Wyboru kombinacji kolorów należy dokonać bardzo ostrożnie, gdyż zbyt duża ich liczba lub ich niezgodność mogą pogorszyć wygląd.

```
FK 1 REM ****************
FJ 2 REM *                  *
BQ 3 REM * Prosram nr 7 *
FL 4 REM *                  *
FO 5 REM ****************
NL 6 REM
FR 40 DIM M$(3),CLS$(1):CLS$=CHR$
   (125):POKE 752,1
VO 45 ? CLS$
YJ 50 POSITION 11,0:? "ROCZNY WYK
   RES CEN"
CD 70 POSITION 6,2:? "    1   2
    3   4   5   6   7   8"
VR 80 POSITION 6,3:? "    0   0
    0   0   0   0   0   0"
IP 90 POSITION 6,4:? "0   0   0
    0   0   0   0   0   0"
NM 100 POSITION 6,5:? "+---+---+-
   --+---+---+---+---+---+"
WY 110 R=(38-6)/(800-0)
HN 120 MIN=10000
BE 130 MAX=0
HN 140 T=0
SV 150 FOR K=1 TO 12
WJ 160 READ M$,S
PB 170 IF S<MIN THEN MIN=S
SZ 180 IF S>MAX THEN MAX=S
HO 190 T=T+S
HC 200 P=INT((S-0)*R+6+0.5)
UB 210 POSITION 0,K+5:? M$;"    :"
   ;:FOR C=7 TO P:? "*";:NEXT C:P
   OSITION 38,K+5:? ":";
YW 215 REM W miejsce "*" mozna ws
   tawic
DI 216 REM znak "U" (CHR$(21))
GQ 220 NEXT K
IV 230 POSITION 6,18:? "+---+---+
   ---+---+---+---+---+---+"
NT 250 POSITION 11,20:? "CENA MIN
   IMALNA ";:POSITION 29+(3-LEN(S
   TR$(MIN))),20:? MIN;
UH 260 POSITION 11,21:? "CENA MAK
   SYMALNA ";:POSITION 29+(3-LEN(
   STR$(MAX))),21:? MAX;
KX 270 POSITION 11,23:? "CENA SRE
   DNIA ";:POSITION 29+(3-LEN(STR
   $(INT(T/12)))),23:? INT(10*T/1
   2)/10;
PR 280 GOTO 280
AX 290 DATA STY,210,LUT,155,MAR,9
   9,KWI,150,MAJ,175,CZE,252,LIP,
   510,SIE,590,WRZ,600,PAZ,751,LI
   S,724,GRU,726

FK 1 REM ****************
FJ 2 REM *                  *
CK 3 REM * Prosram nr 8 *
FL 4 REM *                  *
FO 5 REM ****************
NL 6 REM
AD 60 DIM X(12),Y(12),T$(40),TX$(
   21)
NT 80 GRAPHICS 8+16:COLOR 1
CW 100 T$="WYKRES PRODUKCJI":X=15
   :Y=1:GOSUB 10000
ZK 110 PLOT 80,20:DRAWTO 80,163
NL 120 PLOT 304,163:DRAWTO 304,20
QE 125 PLOT 80,163:DRAWTO 304,163
XT 130 PLOT 304,20:DRAWTO 80,20
TD 140 FOR Y=27 TO 155 STEP 8
AR 150 PLOT 78,Y:DRAWTO 82,Y
MH 160 NEXT Y
RA 170 REM
AP 180 Y=19
WL 190 FOR S=0 TO 800 STEP 100
XE 200 T$="    ":T$(4-LEN(STR$(S))
   )=STR$(S)
AP 210 X=6:GOSUB 10000
YO 220 Y=Y-2
JU 230 NEXT S
QV 240 REM
OA 250 FOR CO=12 TO 34 STEP 2
XY 260 X=CO:Y=20:T$="+"
SP 270 GOSUB 10000
LL 280 NEXT CO
OZ 290 T$="S L M K M C L S W P L
   G":X=12:Y=21:GOSUB 10000
IC 300 T$="T U A W A Z I I R A I
   R":X=12:Y=22:GOSUB 10000
AO 310 T$="Y T R I J E P E Z Z S
   U":X=12:Y=23:GOSUB 10000
IY 320 TX$="WARTOSC PRODUKCJI"
SI 330 FOR K=1 TO 17:T$=TX$(K,K)
BM 340 X=2:Y=K+2:GOSUB 10000
GX 350 NEXT K
IG 360 REM WYKRESLANIE SLUPKOW
ZX 370 T=(155-27)/(800-0)
TQ 380 X1=96
TF 390 FOR K=1 TO 12
ZL 420 READ S
KU 430 Y=INT((800-S)*T+27+0.5)
SI 440 FOR X=X1 TO X1+8
XB 450 PLOT X,Y:DRAWTO X,155
MA 460 NEXT X
LP 470 X1=X1+16
HG 490 NEXT K
LU 500 DATA 210,150,99,250,183,35
   2,410,390,300,651,724,516
NL 510 GOTO 510
WV 9999 REM ZNAKI W GRAFICE 8
ZT 10000 START=PEEK(89)*256+PEEK(
   88)+40*Y*8+X
YC 10010 FOR E1=1 TO LEN(T$):E3=A
   SC(T$(E1))
EL 10020 E3=E3+64*(E3<32)-32*(E3>
   31 AND E3<96)
MK 10030 ZN=PEEK(756)*256+E3*8
EA 10040 FOR E2=7 TO 1 STEP -1
VW 10050 POKE START+E2*40,PEEK(ZN
   +E2):NEXT E2
SI 10060 X=X+1:IF X>=40 THEN STAR
   T=START+40*8:X=0
BT 10070 START=START+1:NEXT E1:RE
   TURN

FK 1 REM ****************
FJ 2 REM *                  *
DE 3 REM * Program nr 9 *
FL 4 REM *                  *
FO 5 REM ****************
NL 6 REM
CY 10 REM ********************
CV 20 REM *KRESKOWANE WYKRESY*
AZ 30 REM *   W GRAFICE 8    *
YM 40 REM *Z OPISEM TEKSTOWYM*
DC 50 REM ********************
VP 60 DIM T$(40),TX$(21)
NT 80 GRAPHICS 8+16:COLOR 1
VD 100 T$="PRODUKCJA GLOBALNA":X=
   15:Y=1:GOSUB 10000
AM 110 PLOT 80,24:DRAWTO 80,162
RL 120 PLOT 304,162:DRAWTO 304,24
OL 125 PLOT 80,162:DRAWTO 304,162
EJ 130 PLOT 304,24:DRAWTO 80,24
TV 140 FOR Y=36 TO 156 STEP 8
AR 150 PLOT 78,Y:DRAWTO 82,Y
MH 160 NEXT Y
RA 170 REM
AP 180 Y=19
VY 190 FOR S=0 TO 30 STEP 10
XE 200 T$="  . ":T$(4-LEN(STR$(S))
   )=STR$(S)
AP 210 X=6:GOSUB 10000
ZP 220 Y=Y-5
JU 230 NEXT S
GK 290 X=0:Y=21:T$="KWARTAL":GOSU
   B 10000
VL 300 X=13:Y=21:T$="1      2
    3      4":GOSUB 10000
IW 310 TX$="WARTOSC PRODUKCJI"
PU 320 FOR K=1 TO 17:T$=TX$(K,K):
   X=1:Y=K+2:GOSUB 10000:NEXT K
IM 330 TX$="W MILIONACH ZL."
UE 340 FOR K=1 TO 15:T$=TX$(K,K):
   X=3:Y=K+4:GOSUB 10000:NEXT K
DA 370 X=35:Y=8:T$="W1":GOSUB 100
   00
SR 380 X=35:Y=13:T$="W2":GOSUB 10
   000
WA 390 X=35:Y=18:T$="W3":GOSUB 10
   000
II 410 T=(156-36)/(30-0)
ZL 420 XL=96
BG 430 XR=XL+32
QC 440 FOR Q=1 TO 4
PN 450 YB=156
DS 500 A=0
KU 510 FOR D=1 TO 3
ZM 520 READ S
BM 540 S=S/10^6
UL 550 YT=INT((30-S)*T+36+0.5)
PA 580 YT=YT-A
KX 590 PLOT XL,YT:DRAWTO XL,YB
SI 600 PLOT XR,YT:DRAWTO XR,YB
IO 610 PLOT XL,YT:DRAWTO XR,YT
UQ 630 FOR X1=XR TO XL STEP -D*3
II 640 X=X1
TQ 650 Y=YT
TJ 660 PLOT X,Y
YD 670 Y=Y+1
YJ 680 X=X-1
QT 690 IF Y<=YB AND X>XL THEN 660
GK 700 NEXT X1
QP 720 FOR Y1=YT TO YB STEP D*3
IT 730 Y=Y1
SN 740 X=XR
TI 750 PLOT X,Y
YC 760 Y=Y+1
YI 770 X=X-1
QR 780 IF Y<=YB AND X>XL THEN 750
HM 790 NEXT Y1
CX 810 A=156-YT
DT 820 YB=YT-1
EG 830 NEXT D
DH 840 XL=XL+48
BO 850 XR=XL+32
JO 870 NEXT Q
MZ 880 DATA 7000000,11000000,4000
   000
XB 890 DATA 8800000,10500000,7000
   000
VN 900 DATA 4000000,12000000,8500
   000
GZ 910 DATA 7000000,11333000,1050
   0000
PQ 920 GOTO 920
WV 9999 REM ZNAKI W GRAFICE 8
ZT 10000 START=PEEK(89)*256+PEEK(
   88)+40*Y*8+X
YC 10010 FOR E1=1 TO LEN(T$):E3=A
   SC(T$(E1))
EL 10020 E3=E3+64*(E3<32)-32*(E3>
   31 AND E3<96)
GP 10025 IF E3=0 THEN 10060
MK 10030 ZN=PEEK(756)*256+E3*8
EA 10040 FOR E2=7 TO 1 STEP -1
VW 10050 POKE START+E2*40,PEEK(ZN
   +E2):NEXT E2
SI 10060 X=X+1:IF X>=40 THEN STAR
   T=START+40*8:X=0
BT 10070 START=START+1:NEXT E1:RE
   TURN
```

W wykresach kolorowych i czarno-białych można stosować różnorodne warianty kreskowania. Podobnie, jak i przy wyborze kolorów, sposób kreskowania należy wybrać w taki sposób aby nie pogorszyć wyglądu rysunku. Jeśli kreskowane są sąsiednie obszary, najbardziej efektywne jest stopniowe zwiększanie kroku kreskowania. Zgodnie z tym zaleceniem tworzony jest histogram za pomocą programu nr 9.

Wykresy punktowe, liniowe lub słupkowe umożliwiają przedstawienie zależności między poszczególnymi wartościami. Niekiedy jednak chcemy pokazać, jakie są relacje wartości zarówno między sobą, jak i względem ich wartości sumarycznej, czyli zależności względne. Można to zrobić za pomocą wykresu słupkowego, podobnie jak w programie nr 9. Jednak znacznie wygodniejsze są diagramy kołowe, szczególnie do przedstawiania informacji o zależnościach procentowych. Poszczególne wartości przedstawiane są jako wycinki koła (sektory). Pole powierzchni takiego wycinka tak się ma do pola powierzchni koła, jak przedstawiana wartość do sumy prezentowanych wartości. Istnieje jednak pewne ograniczenie. Aby diagram był poglądowy, nie powinien mieć więcej niż 6 – 8 sektorów.

Podczas tworzenia diagramów kołowych podstawowym składnikiem programu jest podprogram kreślenia okręgu. Jego szybkość działania decyduje o szybkości kreś-

lenia diagramu. W programie nr 10 wykorzystano jedno z wielu możliwych rozwiązań opisanych w „IKS-ie" nr 5 z 1987 roku.

Odrębnym zadaniem jest pozycjonowanie tekstu nazwy sektora na diagramie. Realizuje się to poprzez wyznaczenie dwusiecznej kąta każdego sektora. Jeśli diagram umieszczony jest w prawej części koła, początek nazwy będzie umieszczony na dwusiecznej, w odległości czterech kroków rastra od okręgu. Jeśli sektor leży w lewej części diagramu, jego nazwa kończy się na dwusiecznej. Początkową pozycję wyświetlania nazwy określa się za pomocą operacji przekształcania współrzędnej punktu w numer pozycji znaku. W tym celu współrzędne X i Y dzieli się przez liczbę elementów rastra przypadających odpowiednio w poziomie i pionie na 1 znak.

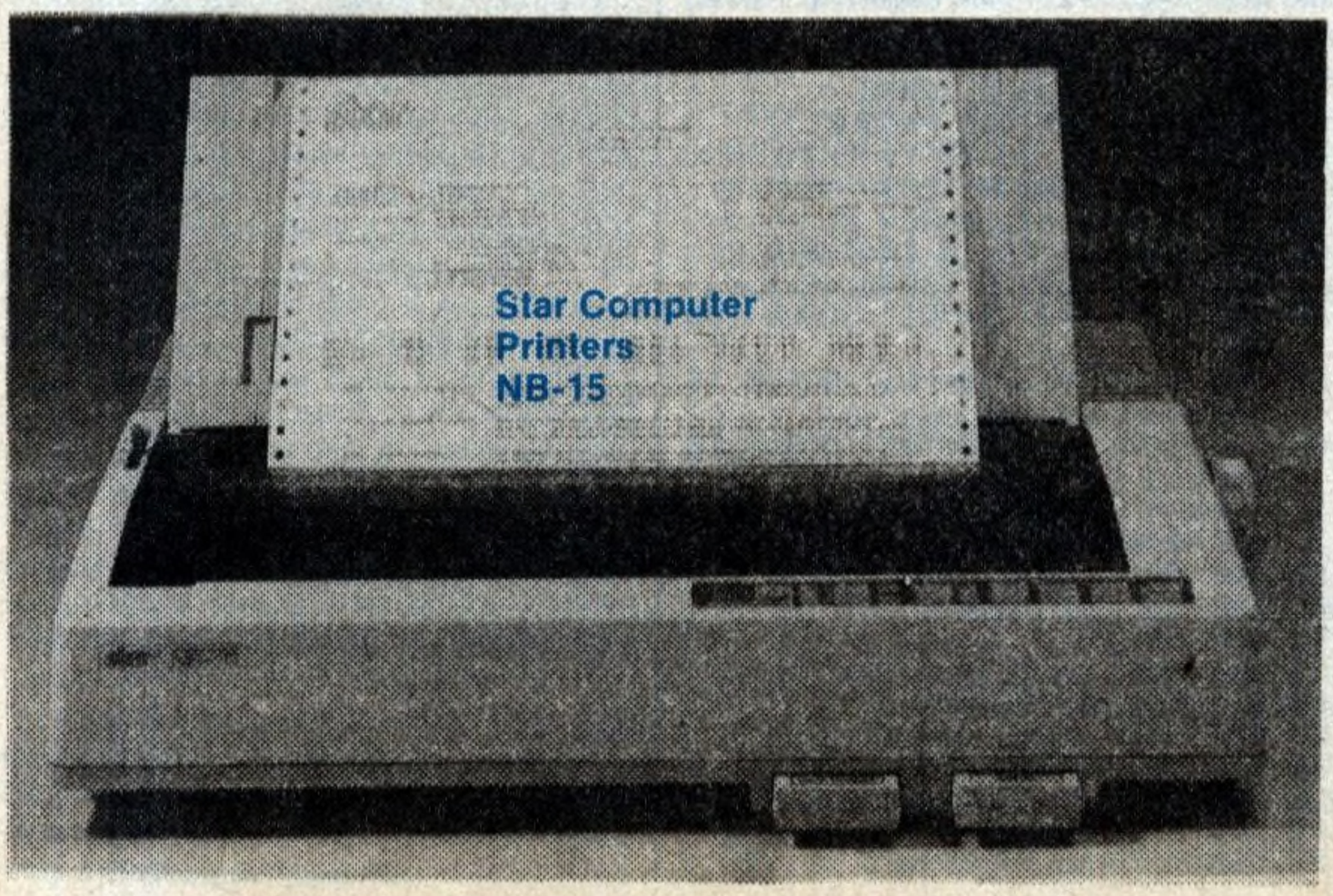
```
QO 1 REM ******************
IB 2 REM *                 *
MQ 3 REM * Prosram nr 10 *
ID 4 REM *                 *
QS 5 REM ******************
NL 6 REM
LQ 20 GRAPHICS 0
QX 30 DIM N$(100),V(8),T$(40),X$(
   40),D(8)
BP 40 XM=320:YM=192
LH 100 C1=40
EE 110 PC=XM/C1
GK 150 PR=8
CQ 160 ?
SG 170 ? "PODAJ WSPOLRZEDNE SRODK
   A WYKRESU"
IF 180 ? "X=";:INPUT XC
JI 190 ? "Y=";:INPUT YC
VV 200 ? "PODAJ PROMIEN WYKRESU "
   ;:INPUT R
NE 210 IF XC+R>XM OR XC-R<0 OR YC
   +R>YM OR YC-R<0 THEN 830
CJ 220 ?
TG 230 ? "PODAJ ILOSC DANYCH (MAX
    8)"
ON 240 INPUT N
OL 250 ? "PODAJ NAZWE WYKRESU"
HV 260 INPUT T$
BV 270 ? "PODAJ NAZWE I WARTOSC "
HW 280 T=0
RQ 290 L=1:REM WPROWADZANIE NAZW
   I WARTOSCI DANYCH
DR 300 FOR K=1 TO N
HA 310 ? "NAZWA ";K;:INPUT X$:? "
   WARTOSC ";:INPUT V:N$(L)=X$:D(
   K)=LEN(X$):X$="":V(K)=V
FH 320 T=T+V(K):L=L+D(K)
GT 330 NEXT K
UP 350 GRAPHICS 8+16:COLOR 1
CL 360 X=INT(C1/2-0.5*LEN(T$)):Y=
   1:REM CENTROWANIE TYTULU
AC 370 GOSUB 10000:REM WYPISANIE
   TYTULU
GK 380 GOSUB 11000:REM KRESLENIE
   OKREGU
CL 390 YA=1
DW 400 B=0
HF 410 S=0
GD 420 RE=360*3.14159/180
CO 425 LL=1
DY 430 FOR K=1 TO N
WB 460 S=S+V(K)
JW 490 A=RE*S/T
SI 500 XP=XC+R*COS(A)
QL 510 YP=YC+R*SIN(A)*YA
PQ 520 PLOT XC,YC:DRAWTO XP,YP
MW 560 AC=B+(A-B)/2
ZP 570 XL=XC+(R+4)*COS(AC)
KE 580 YL=YC+(R+4)*SIN(AC)*YA

OY 590 REM (XL,YL) JEST PUNKTEM
XO 600 REM OKRESLAJACYM START NAZ
   WY JESLI JEST ONA Z PRAWEJ STR
   ONY OKREGU
OY 610 REM KONIEC NAZWY JESLI Z L
   EWEJ
WV 620 REM SRODEK NAZWY JESLI NA
   DOLE LUB GORZE OKREGU
HN 630 REM PUNKT POCZATKOWY
XX 640 IF XL>XC+10 THEN 740
BY 650 REM PUNKT KONCOWY
VQ 660 IF XL<XC-10 THEN 710
WN 670 REM PUNKT SRODKOWY
TA 690 XL=XL-D(K)/2*PC
PM 700 GOTO 740
QU 710 REM
YE 730 XL=XL-D(K)*PC
DZ 740 REM KONWERSJA PUNKTU GR.8
   NA POZYCJE ZNAKU
PS 760 RO=INT(YL/PR+0.5)
DF 770 CO=INT(XL/PC+0.5)
DE 780 X=CO:Y=RO:T$=N$(LL,LL-1+D(
   K)):LL=LL+D(K)
SY 790 GOSUB 10000
IP 800 B=A
GU 810 NEXT K
PF 820 GOTO 820
RE 830 GRAPHICS 0:? "ZLE WSPOLRZE
   DNE !!":FOR I=1 TO 1000:NEXT I
   :RUN
WV 9999 REM ZNAKI W GRAFICE 8
ZT 10000 START=PEEK(89)*256+PEEK(
   88)+40*Y*8+X
YC 10010 FOR E1=1 TO LEN(T$):E3=A
   SC(T$(E1))
EL 10020 E3=E3+64*(E3<32)-32*(E3>
   31 AND E3<96)
GP 10025 IF E3=0 THEN 10060
MK 10030 ZN=PEEK(756)*256+E3*8
EA 10040 FOR E2=7 TO 1 STEP -1
VW 10050 POKE START+E2*40,PEEK(ZN
   +E2):NEXT E2
SI 10060 X=X+1:IF X>=40 THEN STAR
   T=START+40*8:X=0
BT 10070 START=START+1:NEXT E1:RE
   TURN
UE 10939 REM PODPROGRAM KRESLENIA
    OKREGU
HO 11000 FI=0:Y1=0:X1=R
LQ 11010 FIY=FI+Y1+Y1+1
HO 11020 FIXY=FIY-X1-X1+1
BE 11030 PLOT XC+X1,YC+Y1:PLOT XC
   -X1,YC+Y1
FS 11040 PLOT XC+X1,YC-Y1:PLOT XC
   -X1,YC-Y1
BA 11050 PLOT XC+Y1,YC+X1:PLOT XC
   -Y1,YC+X1
FO 11060 PLOT XC+Y1,YC-X1:PLOT XC
   -Y1,YC-X1
HO 11070 FI=FIY:Y1=Y1+1

JY 11080 IF ABS(FIXY)<ABS(FIY) TH
   EN FI=FIXY:X1=X1-1
JS 11090 IF X1>=Y1 THEN 11010
DC 11100 RETURN
```

W celu wyróżnienia jednego lub kilku segmentów diagramu kołowego, przemieszcza się je na pewną odległość od środka, wzdłuż promienia. Tworzenie takiego diagramu, wykorzystane w programie nr 11, opiera się na metodzie pozycjonowania napisów na diagramach kołowych (opisanej powyżej). Najpierw określamy granice kątowe wydzielanego sektora. Następnie obliczamy kąt dla promienia dzielącego ten sektor na połowę. Określamy nowe położenie środka wydzielonego sektora przemieszczonego wzdłuż tego promienia względem środka diagramu. W końcu kreślimy wysunięty sektor. W programie nr 11 wydzielany wycinek koła przesuwany jest o 1/5 promienia okręgu.

```
QO 1 REM ******************
IB 2 REM *                 *
NL 3 REM * Program nr 11 *
ID 4 REM *                 *
QS 5 REM ******************
NL 6 REM
LR 30 GRAPHICS 0
SA 40 DIM N$(100),V(8),T$(40),D(8
   )
BS 70 XM=320:YM=192
CS 120 PC=8
GM 160 PR=8
CS 170 ?
QW 180 ? "PODAJ WSPOLRZEDNE SRODK
   A WYKRESU "
BX 190 ? "X =";:INPUT XN:? "Y =";
   :INPUT YN
MJ 200 ? "PODAJ PROMIEN OKREGU ";
   :INPUT R
VZ 220 IF XN+R+R/5>XM OR XN-R-R/5
   <0 OR YN+R+R/5>YM OR YN-R-R/5<.
   0 THEN 1020
CL 230 ?
MZ 240 REM WPROWADZANIE DANYCH
BZ 250 ? "PODAJ ILOSC DANYCH (MAX
   =8)"
OR 260 INPUT N
BV 270 ? "PODAJ NAZWE I WARTOSC "
SF 280 T=0:L=1
DR 300 FOR K=1 TO N
PZ 310 ? "NAZWA ";K;:INPUT T$:? "
   WARTOSC ";:INPUT V:V(K)=V:D(K)
   =LEN(T$):N$(L)=T$:L=L+D(K):T$=
   ""
WE 320 T=T+V(K)
GT 330 NEXT K
XF 340 ? "KTORA WARTOSC WYROZNIC
   (1 -";N;")";
KV 350 INPUT E
HM 360 IF E<1 OR E>N THEN 340
OI 370 REM KONSTRUKCJA WYKRESU
AG 390 GRAPHICS 8+16:COLOR 1:LL=1
BU 400 YA=1
DY 410 B=0
HH 420 S=0
OE 430 RE=2*3.14159
EA 440 FOR K=1 TO N
```



```
PL 450 XC=XN
QA 460 YC=YN
WH 490 S=S+V(K)
JJ 520 A=RE*S/T
CE 530 IF K<>E THEN 660
MS 540 AC=B+(A-B)/2
TO 550 XE=XC+R/5*COS(AC)
OR 560 YE=YC+R/5*SIN(AC)*YA
UF 570 XP=XE+R*COS(B)
SI 580 YP=YE+R*SIN(B)*YA
OJ 590 TRAP 600:PLOT XE,YE:DRAWTO
    XP,YP
SJ 600 XP=XC+R*COS(A)
QM 610 YP=YC+R*SIN(A)*YA
LV 620 TRAP 630:PLOT XC,YC:DRAWTO
    XP,YP
MG 630 XC=XE
MV 640 YC=YE
MN 660 FOR A1=B TO A STEP 1/R
CK 670 XP=XC+R*COS(A1)
IB 680 YP=YC+R*SIN(A1)*YA
PX 690 TRAP 700:PLOT XP,YP
XO 700 NEXT A1
NA 710 TRAP 750:PLOT XC,YC:DRAWTO
    XP,YP
MW 750 AC=B+(A-B)/2
ZP 760 XL=XC+(R+4)*COS(AC)
KE 770 YL=YC+(R+4)*SIN(AC)*YA
YS 830 IF XL>XC+10 THEN 930
WL 850 IF XL<XC-10 THEN 900
TA 880 XL=XL-D(K)/2*PC
QO 890 GOTO 930
QU 900 REM
YE 920 XL=XL-D(K)*PC
RA 930 REM
DR 950 RO=INT(YL/PR)
RE 960 CO=INT(XL/PC)
DE 970 X=CO:Y=RO:T$=N$(LL,LL-1+D(
    K)):LL=LL+D(K)
SY 980 GOSUB 10000
JI 990 B=A
FL 1000 NEXT K
NE 1010 GOTO 1010
XY 1020 GRAPHICS 0:? "ZLA WARTOSC
    WSPOLRZEDNYCH":FOR I=0 TO 100
    0:NEXT I:RUN
WV 9999 REM ZNAKI W GRAFICE 8
ZT 10000 START=PEEK(89)*256+PEEK(
    88)+40*Y*8+X
YC 10010 FOR E1=1 TO LEN(T$):E3=A
    SC(T$(E1))
EL 10020 E3=E3+64*(E3<32)-32*(E3>
    31 AND E3<96)
GP 10025 IF E3=0 THEN 10060
MK 10030 ZN=PEEK(756)*256+E3*8
EA 10040 FOR E2=7 TO 1 STEP -1
VW 10050 POKE START+E2*40,PEEK(ZN
    +E2):NEXT E2
SI 10060 X=X+1:IF X>=40 THEN STAR
    T=START+40*8:X=0
BT 10070 START=START+1:NEXT E1:RE
    TURN
```

Poszczególne segmenty diagramu kołowego mogą być zapełniane kolorem lub kreskowane, zgodnie z zasadami jak dla histogramów. Kreskowanie powinno być proste, a jego dokładność winna stopniowo wzrastać od sektora do sektora.

Proszę zwrócić uwagę, że w większości przedstawianych programów, wyświetlane dane umieszczone były na stałe w programie, w liniach DATA. Nic nie stoi na przeszkodzie, aby czytelnicy zmodyfikowali je tak, żeby dane były wprowadzane bądź z klawiatury, bądź ze zbioru przechowywanego w pamięci zewnętrznej.

W następnych odcinkach tego cyklu zaprezentujemy „prawdziwy" program do kreślenia różnorodnych wykresów, głównie dla tych czytelników, którzy nie mają ochoty na bawienie się w tworzenie własnych programów i wolą skorzystać z gotowych.

Tomasz MROWIEC
Ludwik PIELA

PROGRAM 46 C-64

Słownik

Po uruchomieniu programu i przedstawieniu się jesteśmy pytani o polski odpowiednik słowa angielskiego lub odwrotnie. Wystawiana przez komputer ocena jest surowa, ale zawsze obiektywna. Program może służyć do nauki języka angielskiego, ale nie tylko. Wszystko zależy od tego, jakie wyrazy wpiszemy w liniach danych. Zmniejszenie lub zwiększenie pojemności naszego słownika wymaga tylko nielicznych zmian w programie. Zaprezentowane w programie efekty dźwiękowe również mogą się podobać i skutecznie zachęcać do nauki języków obcych.

```
10 REM SLOWNIK
30 AT$=CHR$(17):FORT=1TO6:AT$=AT$+AT$:NE
XT T:AT$=CHR$(19)+AT$
35 POKE53281,1:POKE53280,14:GOSUB 10000
40 SC=0
50 QN=0
70 DIME$(20)
80 DIMF$(20)
100 FOR J=1TO20
110 READ E$(J)
120 READ F$(J)
130 NEXT J
135 PRINT CHR$(147)
140 INPUT'    JAK MASZ NA IMIE?';N$
142 N$=LEFT$(N$,8)
150 FOR DL=1TO1000:NEXT DL
154 PRINT CHR$(147);CHR$(144)
160 PRINT LEFT$(AT$,2)SPC(5);'WYNIK: ';S
C
170 PRINT LEFT$(AT$,2)SPC(23);'PYTANIE:
';QN
180 R=INT(RND(1)*20)+1

190 IF RND(1)).5 THEN GOTO 330
200 PRINT LEFT$(AT$,7)SPC(8);'JAK JEST P
O ANGIELSKU';
202 PRINT LEFT$(AT$,8)SPC(8);E$(R);'?
  '
205 PRINT LEFT$(AT$,10)SPC(13);
210 INPUT A$
220 QN=QN+1
230 IF LEN(A$)(15 THENA$=A$+' ':GOTO 230

240 IF LEFT$(A$,LEN(F$(R)))=F$(R)THEN GO
TO 280
250 FOR T=80 TO 0 STEP-1:FOR TT=1TO3:POK
ES0+1,TT*T:NEXT TT,T
260 PRINT LEFT$(AT$,13)SPC(10);'NIE ';N$
;' POWINNO BYC'
262 PRINTLEFT$(AT$,15)SPC(14);CHR$(28);F
$(R)
270 GOTO 150
280 FOR T=10 TO 100:FOR TT=1TO2:POKE SO+
1,T*TT:NEXTTT,T:POKE SO+1,0
290 PRINTLEFT$(AT$,13)SPC(15);'DOBRZE!'
300 SC=SC+1
310 IF SC=20 THEN GOTO 410
320 GOTO 150
330 PRINT LEFT$(AT$,7)SPC(8);'JAK JEST P
O POLSKU'
332 PRINTLEFT$(AT$,8)SPC(8);F$(R);'?
  '
335 PRINTLEFT$(AT$,10)SPC(13);
340 INPUT A$
350 QN=QN+1

360 IF LEN(A$)(10 THEN A$=A$+' ':GOTO 36
0
370 IF LEFT$(A$,LEN(E$(R)))=E$(R) THEN G
OTO 280
380 FOR T=200 TO0 STEP -1:POKE SO+1,T:NE
XT T

390 PRINTLEFT$(AT$,13)SPC(10);'NIE ';N$;
' POWINNO BYC '
392 PRINTLEFT$(AT$,15)SPC(15);CHR$(28);E
$(R)
400 GOTO 150
410 POKE 53281,0
420 POKE 53280,0
430 PRINTCHR$(147);CHR$(158)
440 PRINTLEFT$(AT$,8)SPC(6);'GRATULACJE!
 ODPOWIEDZIALES'
450 PRINT LEFT$(AT$,12)SPC(6);' NA 20 PY
TAN'
460 PRINTLEFT$(AT$,16)SPC(6);'    DOBRZE
. '
470 :

480 :
490 :
500 FOR DL=1TO1000:NEXT DL
510 POKE 53281,7
520 POKE 53280,7
530 PRINT CHR$(147)
540 PRINT CHR$(144):END
600 DATA'DRZWI','DOOR'
610 DATA'LAMPA','LAMP'
620 DATA'DOM','HOUSE'
630 DATA'PIES','DOG'
640 DATA'OGROD','GARDEN'
650 DATA'KOT','CAT'
660 DATA'KURTKA','JACKET'
670 DATA'PROSZEK','POWDER'
680 DATA'KAPELUSZ','HAT'
690 DATA'ROWER','BICYCLE'
700 DATA'KAWA','COFFE'
710 DATA'STACJA','STATION'
720 DATA'CHLEB','BREAD'
730 DATA'MLEKO','MILK'
740 DATA'KUBEK','CUP'
750 DATA'KOBIETA','WOMAN'
760 DATA'DROGA','ROAD'
770 DATA'MAPA','MAP'
780 DATA'MORZE','SEA'
790 DATA'BIURKO','DESK'
10000 REM DZWIEK
10010 SO=54272
10020 FOR T=0TO28:POKESO+T,0:NEXT
10040 POKESO+5,9
10050 POKE SO+6,240
10060 POKE SO+24,15
10070 POKE SO+4,17
10080 RETURN
```

T. CISEK

Krzysztof Walczak

Lilavati — cz. 4

Droga do portu

W kolejnej części cyklu poświęconego rozwiązywaniu anegdot matematycznych chcieliśmy zaprezentować zmodyfikowaną historyjkę o nr. 54 w książce „Lilavati" S. Jeleńskiego.
Oto treść zadania:

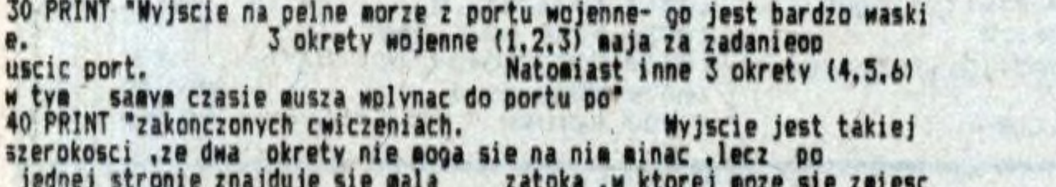
```
Wyjscie na pelne morze z portu wojenne-
go jest bardzo waskie.
3 okrety wojenne (1,2,3) maja za zadanie
opuscic port.
Natomiast inne 3 okrety (4,5,6) w tym
samym czasie musza wplynac do portu po
zakonczonych cwiczeniach.
Wyjscie jest takiej szerokosci ,ze dwa
okrety nie moga sie na nim minac ,lecz
po jednej stronie znajduje sie mala
zatoka ,w ktorej moze sie zmiescic jeden
okret.
Czy okrety wykonaja postawione zadanie?

NACISNIJ 'SPACJE'
```

Tak wygląda sytuacja początkowa:

```
PODAJ NUMER OKRETU: 3

← PORT                    MORZE →
```



Zasady gry:

1. Podawany nr okrętu musi być o wartości 1—6
2. Ruchem okrętu o wprowadzonym numerze steruje się za pomocą klawiszy: ← · → · ↑ · ↓ ·
3. Jeżeli chcemy zmienić nr okrętu, to należy nacisnąć w dowolnej chwili klawisz ENTER.
4. Okręty poruszają się tylko po obszarze niebieskim.
5. Brak możliwości wykonania ruchu jest sygnalizowany dźwiękiem.
6. Gra kończy się sukcesem po wykonaniu zadania.

Segment sterujący przebiegami gry ma następującą postać:

```
10 MODE 1:INK 0,11:INK 1,0:INK 2,26:INK 3,1:PAPER 0:BORDER 11:CLS:PEN
1
20 LOCATE 1,5
30 PRINT "Wyjscie na pelne morze z portu wojenne- go jest bardzo waski
e.                3 okrety wojenne (1,2,3) maja za zadanieop
uscic port.                          Natomiast inne 3 okrety (4,5,6)
w tym  samym czasie musza wplynac do portu po"
40 PRINT "zakonczonych cwiczeniach.                  Wyjscie jest takiej
szerokosci ,ze dwa  okrety nie moga sie na nim minac ,lecz  po
 jednej stronie znajduje sie mala     zatoka ,w ktorej moze sie zmiesc
ic jedenokret."
50 PRINT "Czy okrety wykonaja postawione zadanie?"
60 PEN 2:LOCATE 1,23:PRINT "NACISNIJ 'SPACJE'"
70 IF INKEY$="" THEN GOTO 70
80 MODE 1:INK 0,11:INK 1,0:INK 2,26:INK 3,1:PAPER 0:BORDER 0:CLS
90 GOSUB 1230
100 GRAPHICS PEN 1:PLOT 0,128:DRAWR 640,0:PLOT 0,200:DRAWR 240,0:DRAWR
 32,48:DRAWR 48,0:DRAWR 32,-48:DRAWR 360,0
110 MOVE 10,10:FILL 1:MOVE 10,300:FILL 1
120 PEN 2:PAPER 1:LOCATE 1,10:PRINT  CHR$(242);" PORT"
130 LOCATE 34,10:PRINT  "MORZE ";CHR$(243):PAPER 0
140 DIM wsp(6,2)
150 FOR a=1 TO 3
160 wsp(a,1)=7+(a-1)*3
170 wsp(a,2)=13
180 NEXT a
190 FOR a=4 TO 6
200 wsp(a,1)=26+(a-4)*3
210 wsp(a,2)=13
220 NEXT a
230 FOR K=1 TO 6
240 GOSUB 1210
250 NEXT
260 WINDOW £1,1,40,1,6:PAPER £1,1
270 CLS £1:PEN £1,2:LOCATE £1,2,2:INPUT £1,"PODAJ NUMER OKRETU: ",N
280 IF N>0 AND N<7 THEN GOTO 300
290 SOUND 1,358:SOUND 1,458:CLS £1:PRINT £1,"BLEDNY NUMER OKRETU":FOR
T=1 TO 2000:NEXT:GOTO 270
300 CLS £1:PRINT £1,"STERUJ RUCHEM OKRETU  NR ":USING "£";N;:PRINT £1,
"ZA POMOCA KLAWISZY:";CHR$(240);",";CHR$(241);",";CHR$(242);",
";CHR$(243):LOCATE £1,1,4:PRINT £1,"JEZELI CHCESZ ZMIENIC NR OKRETU NA
CISNIJKLAWISZ 'ENTER'"
310 A$=INKEY$
320 IF A$="" THEN GOTO 310
330 IF INKEY(6)=0 THEN GOTO 270
340 P=ASC(A$)
350 IF p<240 OR p>243 THEN GOTO 310
360 ON P-239 GOSUB 510,720,950,1080
370 FOR G=1 TO 3
380 IF WSP(G,1)<=18 THEN GOTO 500
390 NEXT G
400 FOR G=4 TO 6
410 IF WSP(G,1)>=18 THEN GOTO 500
420 NEXT G
430 CLS £1:PRINT £1,"BRAWO! UDALO SIE!"
440 FOR A=0 TO 26
450 BORDER A
460 FOR T=1 TO 50:NEXT T
470 NEXT A
480 IF INKEY$="" THEN GOTO 440
490 END
500 GOTO 310
```

Oprócz segmentu sterującego inne czynności zorganizowano w postaci podprogramów, których rola jest następująca:

1. Kontrola, czy ruch w górę jest możliwy (instrukcje 510—710).

```
510 IF wsp(n,1)<>18 THEN SOUND 1,358:RETURN
520 IF wsp(n,2)=10 THEN SOUND 1,358:RETURN
530 FOR I=1 TO 6
540 IF WSP(N,2)-1=WSP(I,2)+2 THEN GOTO 660
550 NEXT I
560 IF r1=1 THEN GOTO 580
570 GOSUB 1330
580 X=WSP(N,1):Y=WSP(N,2)+1:GOSUB 1840
590 POKE &7F74,W2:POKE &7F75,W1
600 FOR b=1 TO 3
610 FOR C=1 TO 8
620 POKE &7F02,C:CALL &7F00
630 FOR T=1 TO 2:NEXT:FRAME:NEXT:NEXT
640 WSP(N,2)=WSP(N,2)-3
650 RETURN
660 FOR R=0 TO 2
670 FOR J=0 TO 2
680 IF WSP(N,1)+R=WSP(I,1)+J THEN SOUND 1,358:RETURN
690 NEXT J
700 NEXT R
710 GOTO 550
```

2. Kontrola, czy ruch w dół jest możliwy (instrukcje 720—940).

```
720 IF wsp(n,1)<>18 OR WSP(N,2)<>10 THEN SOUND 1,358:RETURN
730 FOR I=1 TO 6
740 IF WSP(N,2)-3=WSP(I,2) THEN GOTO 880
750 IF n=1 THEN GOTO 770
760 IF WSP(I,1)=18 THEN SOUND 1,358:RETURN
770 NEXT I
780 IF r1=4 THEN GOTO 800
790 GOSUB 1580
800 X=WSP(N,1):Y=WSP(N,2)+3:GOSUB 1840
810 POKE &7F74,W2:POKE &7F75,W1+56
820 FOR b=1 TO 3
830 FOR C=1 TO 8
840 POKE &7F02,C:CALL &7F00
850 FOR T=1 TO 2:NEXT:FRAME:NEXT:NEXT
860 WSP(N,2)=WSP(N,2)+3
870 RETURN
880 FOR R=0 TO 2
890 FOR j=0 TO 2
900 IF WSP(N,1)+R=WSP(I,1)+J THEN SOUND 1,358:RETURN
910 NEXT J
920 NEXT R
930 GOTO 770
940 RETURN
```

3. Kontrola, czy ruch w lewo jest możliwy (instrukcje 950—1070).

```
950 IF WSP(N,2)<13 THEN SOUND 1,358:RETURN
960 IF WSP(N,1)=1 THEN SOUND 1,358:RETURN
970 FOR I=1 TO 6
980 IF WSP(N,1)=WSP(I,1)+3 AND WSP(I,2)=13 THEN SOUND 1,358:RETURN
990 NEXT I
1000 IF r1=3 THEN GOTO 1020
1010 GOSUB 1750
1020 x=wsp(n,1):y=wsp(n,2)+1:GOSUB 1840
1030 POKE &7F7B,w1:FOR z=w2 TO w2-1 STEP -1
1040 FRAME:POKE &7F7A,z::CALL &7F76
1050 NEXT
1060 WSP(N,1)=WSP(N,1)-1
1070 RETURN
```

4. Kontrola, czy ruch w prawo jest możliwy (instrukcje 1080—1200).

```
1080 IF WSP(N,2)<13 THEN SOUND 1,358:RETURN
1090 IF WSP(N,1)=38 THEN SOUND 1,358:RETURN
1100 FOR I=1 TO 6
1110 IF WSP(N,1)+3=WSP(I,1) AND wsp(i,2)=13 THEN SOUND 1,358:RETURN
1120 NEXT I
1130 IF r1=2 THEN GOTO 1150
1140 GOSUB 1490
1150 x=wsp(n,1)+2:y=wsp(n,2)+1:GOSUB 1840
1160 POKE &7F7B,w1:FOR z=w2+1 TO w2+2
1170 FRAME:POKE &7F7A,z::CALL &7F76
1180 NEXT
1190 WSP(N,1)=WSP(N,1)+1
1200 RETURN
```

5. Rysowanie jednego okrętu (instrukcje 1210—1220).

```
1210 FOR a=1 TO 3:PEN 3:LOCATE wsp(k,1),wsp(k,2)+a:PRINT b$(a):PEN 1:L
OCATE wsp(k,1),wsp(k,2)+2:PRINT USING "£":k:PEN 3:NEXT
1220 RETURN
```

6. Definiowanie symboli graficznych okrętu (instrukcje 1230—1320).

```
1230 SYMBOL AFTER 125
1240 SYMBOL 125,0,0,0,1,1,1,1
1250 SYMBOL 126,1,1,1,1,1,1,1
1260 SYMBOL 127,0,64,91,255,219,255,127,127
1270 SYMBOL 128,13,15,235,255,109,255,255,255
1280 SYMBOL 129,96,228,164,252,172,248,248,240
1290 b$(1)=" "+CHR$(125)+" "
1300 b$(2)=" "+CHR$(126)+" "
1310 b$(3)=CHR$(127)+CHR$(128)+CHR$(129)
1320 RETURN
```

7. 4 podprogramy (instrukcje 1330—1830) wczytywania do pamięci podprogramów ruchu w określonym kierunku w języku wewnętrznym:

```
1330 REM  'przesuwanie w gore"
1340 r1=1
1350 RESTORE 1370:FOR i=&7F00 TO &7F73
1360 READ v:POKE i,v:NEXT
1370 DATA 1,24,1,42,116,127,16,93
1380 DATA 229,124,198,56,103,17,80,0,237,82,84
1390 DATA 93,225,197,229,213,205,95,127,209,98,107
1400 DATA 34,116,127,225,193,13,40,46,6,7,84,93,124
1410 DATA 198,8,103,197,229,213
1420 DATA 205,95,127,209,225,193,13,40,26,16,236
1430 DATA 84,93,124,214,56,103,213,17,80,0,237,90
1440 DATA 209,197,229,213,205,95,127,209,225,193
1450 DATA 24,207,54,0,84,93,19,1,5,0,237,176,201
1460 DATA 1,6,0,237,176,201,124,214,8,87
1470 DATA 93,229,98,107,34,116,127,225,24,187,201
1480 RETURN
1490 REM 'przesuwanie w prawo'
1500 r1=2
1510 RESTORE 1530:FOR i=&7F76 TO &7FA1
1520 READ v:POKE i,v:NEXT
1530 DATA 1,8,3,33,39,192,84
1540 DATA 93,19,197,1,6,0,237,184,193,35,54,0
1550 DATA 13,40,6,17,5,8,25,24,234,5,40,12,124
1560 DATA 214,56,103,17,85,0,25,14,8,24,219,201
1570 RETURN
1580 REM 'przesuwanie w dol'
1590 r1=4
1600 RESTORE 1620:FOR i=&7F00 TO &7F73
1610 READ v:POKE i,v:NEXT
1620 DATA 1,24,1,42,116,127,16,93,229
1630 DATA 124,214,56,103,17,80,0,237,90,84
1640 DATA 93,225,197,229,213,205,95,127,209
1650 DATA 98,107,34,116,127,225,193,13,40,46
1660 DATA 6,7,84,93,124,214,8,103,197,229,213
1670 DATA 205,95,127,209,225,193,13,40,26,16,236
1680 DATA 84,93,124,198,56,103,213,17,80,0,237,82
1690 DATA 209,197,229,213,205,95,127
1700 DATA 209,225,193,24,207
1710 DATA 54,0,84,93,19,1,6,0,237,176,201
1720 DATA 1,6,0,237,176,201,124,198,8,87
1730 DATA 93,229,98,107,34,116,127,225,24,187,201
1740 RETURN
1750 REM 'przesuwanie w lewo'
1760 r1=3
1770 RESTORE 1790:FOR i=&7F76 TO &7FA1
1780 READ v:POKE i,v:NEXT
1790 DATA 1,8,3,33,36,192,84,93
1800 DATA 27,197,1,6,0,237,176,193,43,54,0
1810 DATA 13,40,6,17,251,7,25,24,234,5,40,12,124
1820 DATA 214,56,103,17,75,0,25,14,8,24,219,201
1830 RETURN
```

8. Przeliczanie współrzędnych okrętu na adresy ekranu, potrzebne do wymienionych w pkt. 7 podprogramów (instrukcje 1840—1880).

```
1840 A=(Y-1)*80+(x-1)*2
1850 W1=INT(a/256)
1860 W2=a-W1*256
1870 w1=192+w1
1880 RETURN
```

Życzymy przyjemnej zabawy i sukcesów w rozwiązywaniu tego problemu.

Danuta KWASIŻUR
Mieczysław SKONIECZNY

PROGRAM 47

ZX SPECTRUM

Test joystck'a

Po pewnym okresie używania joystick'a dają się zauważyć jego niedomagania. Niektóre z wychyleń przestają funkcjonować poprawnie. Czasem jesteśmy w rozterce: czy jest to wina źle załadowanego programu, czy też samej przystawki. Prezentowany test wyeliminuje te wątpliwości. Należy przyłączyć joystick, uruchomić program, a następnie poruszać dźwignią we wszystkich możliwych kierunkach. Jeśli uznacie, że test trwa zbyt długo, to wystarczy nacisnąć spację z klawiszem CONTROL i otrzymacie informacje o funkcjach, które nie działają poprawnie. Zwykle jest to wina styków, które po rozebraniu joystick'a należy odpowiednio przygiąć i przeczyścić.

Krzysztof MAMCARZ

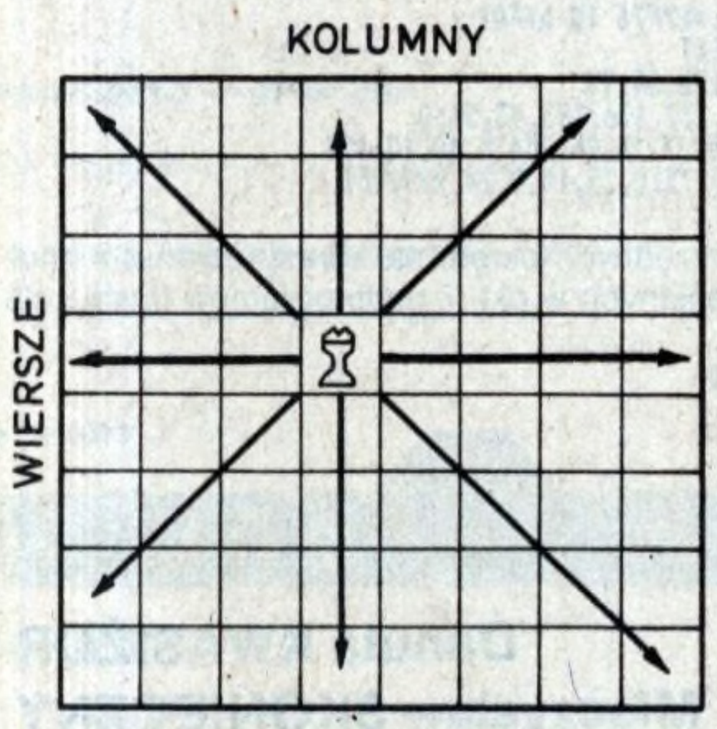
```
10 ' ** testjoy '87
20 ' program do testowania
30 ' joystick'a
40 ' AMSTRAD CPC
50 '
60 MODE 1
70 DEFINT a-z
80 WINDOW #1,1,40,25,25
90 DIM f(8)
100 FOR a=5 TO 15 STEP 5
110 FOR b=5 TO 15 STEP 5
120 LOCATE b,a: PRINT ","
130 NEXT b,a
140 PRINT #1,"przygotuj JOYSTICK i nacis
nij cokolwiek.";
150 CLEAR INPUT: WHILE INKEY$="": WEND
160 PRINT #1, "naciskaj wszystkie poloze
nia joystick'a"
170 IF JOY(0)=1 THEN f(0)=1: LOCATE 10,5
: PRINT " "
180 IF JOY(0)=2 THEN f(1)=1: LOCATE 10,1
5: PRINT" "
190 IF JOY(0)=4 THEN f(2)=1: LOCATE 5,10
: PRINT " "
200 IF JOY(0)=5 THEN f(3)=1: LOCATE 5,5:
 PRINT " "
210 IF JOY(0)=6 THEN f(4)=1: LOCATE 5,15
: PRINT " "
220 IF JOY(0)=8 THEN f(5)=1: LOCATE 15,1
0: PRINT " "
230 IF JOY(0)=9 THEN f(6)=1: LOCATE 15,5
: PRINT " "
240 IF JOY(0)=10 THEN f(7)=1: LOCATE 15,
15: PRINT " "
250 IF JOY(0)=16 THEN f(8)=1: LOCATE 10,
10: PRINT " "
260 IF INKEY(47)=128 THEN 280
270 GOTO 170
280 ' CTRL-SPACE
290 CLS #1: CLEAR INPUT
300 f1=0: LOCATE 1,16
310 FOR a=1 TO 9
320 RESTORE 410
330 IF f(a-1)=0 THEN GOSUB 420
340 NEXT
350 IF f1=1 THEN LOCATE 1,1: PRINT "BRAK
 REAKCJI:" ELSE CLS: PRINT "JOYSTICK JES
T SPRAWNY"
360 LOCATE 20,12: PRINT "ponownie ? T/N"
370 WHILE INKEY$="": WEND
380 IF INKEY(51)>=0 THEN RUN
390 CLS :END
400 '
410 DATA "gora","dol","lewo","lewo-gora"
,"lewo-dol","prawo","prawo-gora","prawo-
dol","fire"
420 FOR b=1 TO a: READ h$:NEXT b
430 PRINT h$
440 f1=1
450 RETURN
```

PROGRAM 48

CPC 6128

Gra w ośmiu hetmanów

Zagadnienie ośmiu hetmanów polega na znalezieniu takiego ich rozmieszczenia na 64-polowej kwadratowej szachownicy, aby żaden hetman nie „bił" innego. Mówiąc dokładniej, żadne dwa hetmany nie mogą znajdować się w tej samej kolumnie, w tym samym wierszu, ani na tej samej przekątnej. Kierunki „bicia" hetmana prezentuje rys. 1.



Rys. 1. Kierunki „bicia" hetmana

Istnieje 92 rozwiązań rozmieszczenia ośmiu hetmanów.

W grze znalezienie poprawnego rozmieszczenia umożliwia przejście na kolejny z poziomów, określone jest ilością możliwych rozwiązań. Utworzenie jednego rozwiązania polega na ustawieniu w kolejnych kolumnach (początek w pierwszej) szachownicy hetmanów nie „bijących się" wzajemnie. Ustawienie zadawane jest poprzez numer wiersza. Numery kolumn ustawiane są automatycznie. Każde poprawne ustawienie hetmana jest rysowane na ekranie oraz zwiększa liczbę punktów o 25.

Próba błędnego ustawienia — hetman jest „bity" przez już ustawionego hetmana — powoduje utratę 25 punktów. Tak więc na jednym poziomie można uzyskać 200 punktów. Stracić można więcej, gdyż można doprowadzić do sytuacji, w której niemożliwe jest ustawienie ośmiu hetmanów. Wznowienie gry następuje wtedy na tym samym poziomie. Grę kończy ustawienie 92 różnych rozmieszczeń ośmiu hetmanów, lub zniechęcenie gracza, któremu po prawidłowym lub błędnym rozmieszczeniu dostarczana jest taka możliwość. Kolorowa grafika oraz efekty dźwiękowe powinny sprzyjać znajdowaniu prawidłowych rozmieszczeń hetmanów na szachownicy.

Gra ośmiu hetmanów napisana została w języku PASCAL oraz uruchomiona na komputerze AMSTRAD CPC 6128. Program źródłowy gry tworzą dwa zbiory. Zbiór główny zawiera procedury graficzne oraz moduł sterujący całego programu, z którego wywoływane są dwie procedury umieszczone w drugim zbiorze źródłowym gry. Jedna z tych procedur obrazuje czołówkę gry. Druga zawiera demonstrację gry oraz samą grę. Z uwagi na obszerność tych procedur oraz ich wzajemne wykluczanie się, zostały one zaimplementowane w postaci procedur nakładkowych.

URUCHOMIENIE PROGRAMU

Po przepisaniu programu źródłowego do dwóch zbiorów źródłowych o nazwach OSIEMHET. PAS i PROCPOM. PAS należy wywołać opcję kompilatora języka TURBO PASCAL — komenda O. Następnie użyć komendy C (kompilacja do zbioru) oraz Q (koniec opcji). Kolejnym krokiem jest wywołanie komendy M oraz wpisanie nazwy zbioru OSIEMHET. PAS. Kompilację uruchamia komenda C. W wyniku jej realizacji utworzone zostaną na dyskietce dwa zbiory o nazwie OSIEMHET. COM i OSIEMHET. 000. Od tej chwili uruchomienie gry jest możliwe pod systemem CP/M PLUS.

Życzymy satysfakcji w osiąganiu kolejnych rozwiązań, z którymi kłopoty miał nawet C. F. GAUSS.

Andrzej CETERA

```
PROGRAM osiemhetmanow;
{         PROGRAM GLOWNY            }
{$a-}
{ ################################## }
{       GRAFIKA TURBO PASCALA       }
{ ################################## }

const CPM3 = true;

const tagflag:boolean = false;
      tagaddr:integer = 0;
      indscr :integer = 0;
      fct            = 0.017453292520;
      xerror :real    = 0.0;
      yerror :real    = 0.0;
      getmode:byte    = 2;
      getgrafmode:byte= 0;

var   itg:integer;
      btg:byte;
      rd:array[1..13] of byte;
      re:real;

procedure initgrafik;
begin
   write(#$1b,'0',#$1b,'y');
end;

procedure leavegrafik;
begin
   write(#$1b,'1',#$1b,'x');
end;

procedure gotoxy(x,y:integer);
begin
   write(#$1b,'Y',chr(y+31),chr(x+31));
end;

procedure
sound(g:byte;h,i:integer;j,k,l,m:byte);
var sd:array[0..8] of byte;
begin
  sd[0] := g;
  move(h,sd[3],2);
  move(i,sd[7],2);
  sd[1] := k;
  sd[2] := l;
  sd[5] := m;
  sd[6] := j;
  inline($21/sd/$cd/$5a/$fc/$aa/$bc);
end;

procedure soundreset;
begin
  inline($cd/$5a/$fc/$a7/$bc);
end;
procedure draw(x,y:integer);
begin
inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/$F6/
       $BB);
end;
procedure drawr(x,y:integer);
begin
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/
         $F9/$BB);
end;

procedure grafmove(x,y:integer);
begin
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/
         $C0/$BB);
end;

procedure grafmover(x,y:integer);
begin
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/
         $c3/$BB);
end;

procedure plot(x,y:integer);
begin
  inline($2A/x/$EB/$2A/y/$cd/$5a/$fc/
         $EA/$BB);
end;

procedure
grafwindow(x1,y1,x2,y2:integer);
begin
  inline($2A/x2/$EB/$2A/x1/$cd/$5a/$fc/
         $CF/$BB/
         $2A/y2/$EB/$2A/y1/$cd/$5a/$fc/
         $D2/$BB);
end;

procedure grafpen(fs:byte);
begin
  inline($3A/fs/$cd/$5a/$fc/$DE/$BB);
end;

procedure grafpaper(fs:byte);
begin
  inline($3A/fs/$cd/$5a/$fc/$E4/$BB);
end;

procedure clg(fs:byte);
begin
  inline($cd/$5a/$fc/$DB/$BB);
  grafpaper(fs);
end;

procedure ink(fs,f1,f2:byte);
begin
  inline($3a/f1/$47/$3a/f2/$4f/$3a/fs/
         $cd/$5a/$fc/
         $32/$bc);
end;

procedure paper(fs:byte);
begin
  inline($3A/fs/$cd/$5a/$fc/$96/$BB);
end;

procedure pen(fs:byte);
begin
  inline($3A/fs/$cd/$5a/$fc/$90/$BB);
end;

procedure mode(md:byte);
begin
  if md in [0..2] then
  begin
    getmode := md;
    inline($3a/md/$cd/$5a/$fc/$0e/$bc);
  end;
end;

procedure border(v1,v2:byte);
begin
  inline($3a/v1/$47/$3a/v2/$4f/$cd/$5a/
         $fc/$38/$bc);
end;

procedure grafout(ch:char);
begin
  inline($3A/ch/$cd/$5a/$fc/$FC/$BB);
end;

procedure tag;
begin
  if not tagflag then
  begin
    tagaddr := conoutptr;
    tagflag := true;
    conoutptr := addr(grafout);
  end;
end;

procedure tagoff;
begin
  if tagflag then
  begin
    conoutptr := tagaddr;
    tagflag := false;
  end;
end;
{$iprocpom.pas}
{########################################}
{          MODUL STERUJACY               }
{########################################}
begin
crtinit;
initgrafik;
czolowka;
g8h;
mode(2);
grafwindow(1,1,1,1);
gotoxy(1,1);
ink(0,26,26);
ink(1,1,1);
paper(1); pen(0);
border(1,1);
leavegrafik;
crtexit;
end.
```

```
{########################################}
{  PROCEDURA CZOLOWKA                    }
{########################################}
overlay
procedure czolowka;

var wsp:array [1..16,1..3] of real;
    wsp1:array [1..16,1..3] of real;
    nwsp:array [1..16,1..3] of real;
     mp:array [1..3,1..3] of real;
     ms:array [1..3,1..3] of real;
     pw:array [1..16,1..2] of real;
     fla,licz,k,nh:integer;
   sx,sy,rx,ry,xp,yp,krok:real;

procedure tytul;
var t:string[18]; j:integer;
begin
t:=' 8 H E T M A N O W';
grafpen(9); tag;
plot(40,375);
for j:=1 to 18 do
grafout(t[j]);
tagoff;
end;

procedure podp;
var t:string[15]; j:integer;
begin
t:='wcisnij klawisz';
grafpen(10); tag;
plot(100,20);
for j:=1 to 15 do
grafout(t[j]);
tagoff;
end;
{########################################}
{  PROCEDURY PRZESUWANIA I WYPELNIANIA   }
{########################################}

procedure  mnoz(i:integer);
var n,k,m:integer;
begin
for n:=1 to i do
for k:=1 to 3 do
for m:=1 to 3 do
  nwsp[n,k]:=nwsp[n,k]+wsp[n,m]*ms[m,k];
end;

procedure  mnoz1(i:integer);
var n,k,m:integer;
begin
for n:=1 to i do
for k:=1 to 3 do
for m:=1 to 3 do
  nwsp[n,k]:=nwsp[n,k]+wsp[n,m]*mp[m,k];
end;
procedure  ustms(i:integer);
var n,k:integer;
begin
ms[1,2]:=0.0;
ms[1,3]:=0.0;
ms[1,1]:=sx;
ms[2,1]:=0.0;
ms[2,2]:=sy;
ms[2,3]:=0.0;
ms[3,1]:=-rx*sx+rx;
ms[3,2]:=-ry*sy+ry;
ms[3,3]:=1.0;
for k:=1 to i do
for n:=1 to 3 do
 nwsp[k,n]:=0.0;
end;
procedure  ustmp(i:integer; p,q:real);
var n,k:integer;
begin
mp[1,2]:=0.0;
mp[1,3]:=0.0;
mp[1,1]:=1.0;
mp[2,1]:=0.0;
mp[2,2]:=1.0;
mp[2,3]:=0.0;
mp[3,1]:=p;
mp[3,2]:=q;
mp[3,3]:=1.0;

for k:=1 to i do
for n:=1 to 3 do
 nwsp[k,n]:=0.0;
end;

procedure rysno(i:integer);
var n:integer;
begin
plot(round(nwsp[1,1]),round(nwsp[1,2]));
for n:=2 to i do
begin
draw(round(nwsp[n,1]),round(nwsp[n,2]));
sound(1,round(nwsp[n,1])-100,2,2,0,0,0);
sound(2,round(nwsp[n,2]),2,2,0,0,0);
end;
end;

{########################################}
{       INICJACJA DANYCH                 }
{########################################}

procedure inicda;
begin
fla:=0;
pw[1,1]:=115.0;
pw[1,2]:=0.0;
pw[2,1]:=115.0;
pw[2,2]:=5.0;
pw[3,1]:=120.0;
pw[3,2]:=5.0;
pw[4,1]:=125.0;
pw[4,2]:=20.0;
pw[5,1]:=120.0;
pw[5,2]:=30.0;
pw[6,1]:=120.0;
pw[6,2]:=35.0;
pw[7,1]:=125.0;
pw[7,2]:=40.0;
pw[8,1]:=130.0;
pw[8,2]:=35.0;
pw[9,1]:=135.0;
pw[9,2]:=40.0;
pw[10,1]:=140.0;
pw[10,2]:=35.0;
pw[11,1]:=140.0;
pw[11,2]:=30.0;
pw[12,1]:=135.0;
pw[12,2]:=20.0;
pw[13,1]:=140.0;
pw[13,2]:=5.0;
pw[14,1]:=145.0;
pw[14,2]:=5.0;
pw[15,1]:=145.0;
pw[15,2]:=0.0;
pw[16,1]:=115.0;
pw[16,2]:=0.0;
mode(0);
grafwindow(0,0,640,400);
ink(0,0,0);
ink(1,6,6);
ink(2,9,9);
ink(3,26,26);
ink(4,7,7);
ink(5,16,16);
ink(6,11,11);
ink(7,15,15);
ink(8,23,23);
ink(9,12,12);
ink(10,25,25);
grafpaper(0); border(0,0);
clg(0);
end;

procedure pdan(nr:integer);
var i:integer;
begin
for i:=1 to 16 do
begin
wsp[i,1]:=pw[i,1];
wsp[i,2]:=pw[i,2] +nr*40;
wsp[i,3]:=1;
end;
xp:=130.0;
yp:=20.0+nr*40.0;
krok:=40.0;
rx:=xp;
ry:=yp;
licz:=16;
end;

procedure pdan1(i:integer);
var k,n:integer;
begin
for k:=1 to i do
for n:=1 to 3 do
  wsp1[k,n]:=wsp[k,n];
end;
{########################################}
{     PROCEDURY POMOCNICZE               }
{########################################}

procedure przep1(i:integer);
var k,n:integer;
begin
for k:=1 to i do
begin
for n:=1 to 2 do
  wsp[k,n]:=wsp1[k,n];
wsp[k,3]:=1;
end;
end;

procedure przep2(i:integer);
var k,n:integer;
begin
for k:=1 to i do
begin
for n:=1 to 2 do
  wsp[k,n]:=nwsp[k,n];
wsp[k,3]:=1;
end;
end;

procedure rysfig(i,nr:integer);
var x:integer;
begin
 grafpen(nr);
 sx:=1.0;
 sy:=1.0;
for x:=1 to 12 do
 begin
   ustms(i);
   mnoz(i);
   rysno(i);
   przep2(i);
   sy:=sy-0.02;
   sx:=sx-0.02;
 end;
 end;
{########################################}
{  MODUL STERUJACY PROCEDURY CZOLOWKA    }
{########################################}
begin
inicda;
repeat
 if (fla=0) then
 nh:=1 else
 begin nh:=random(7)+1; tytul; podp;
 end;
repeat
 pdan(nh); pdan1(licz);
 rysfig(licz,nh);
 for k:=1 to 8-random(7) do
    begin
     przep1(licz); {odtworzenie}
     rx:=rx+ krok;
     ustmp(licz,krok,0);
     mnoz1(licz);
     przep2(licz); pdan1(licz);
     grafwindow(0,nh*40,640,40+nh*40);
     clg(0);
     grafwindow(0,0,640,400);
     rysfig(licz,nh);
    end;
 nh:=nh+1;
until (nh=9) or (fla=1);
  fla:=1;
until keypressed;
border(0,3);
end;

{########################################}
{       PROCEDURA GRA                    }
{########################################}
overlay
procedure g8h;
var i,poz,ilpun:integer; w:boolean;
    nr,m,n:integer;
    tuw:array [1..92,1..8] of integer;
    a:array [1..8] of boolean;
    b:array [2..16] of boolean;
    c:array [-7..7] of boolean;
    x:array [1..8] of integer;
    co,y,zc:char;
const
    poczgcy=390;
    poczgcx=100;
    poczbcy=363;
    poczbcx=68;
    poczl=86;
    dlink=288;
    dlh=36;
    pxh=50;
    pyh=374;

{########################################}
{   PROCEDURY GRAFICZNE GRY              }
{########################################}
procedure cyfra(ko:integer);
 begin
  case ko of
  1:zc:='1';
  2:zc:='2';
  3:zc:='3';
  4:zc:='4';
  5:zc:='5';
  6:zc:='6';
  7:zc:='7';
  8:zc:='8';
  end;
 end;

procedure rysszach;
var j:integer;
 begin
  grafpen(2);
  grafpaper(0);
  for j:=0 to 8 do
   begin
    plot(poczl,j*36+poczl);
    drawr(dlink,0);
   end;
  for j:=0 to 8 do
   begin
    plot(j*36+poczl,poczl);
    drawr(0,dlink);
   end;
  grafpen(3);
  tag;
  plot(poczgcx,poczgcy);
  for j:=1 to 8 do
   begin
     cyfra(j);
     grafout(zc);
     grafmove(poczgcx+j*36,poczgcy);
   end;
  plot(poczbcx,poczbcy);
  for j:=1 to 8 do
   begin
     cyfra(j);
     grafout(zc);
     grafmove(poczbcx,poczbcy-j*36);
   end;
  tagoff;
end;

procedure ryshet(k:integer);
var nw,j:integer;
begin
 nw:=x[k];
 grafpen(3);
 for j:=0 to 35 do
  begin
   plot(pxh+k*36,pyh-nw*36+j);
   drawr(dlh,0);
   sound(1,100-j*3,5,5,0,0,0);
   sound(2,150-j*4,5,5,0,0,0);
  end;
 grafpen(2);
 plot(pxh+k*36,pyh-nw*36);
 grafmove(5,4);
 drawr(6,18); drawr(-6,9);
 drawr(9,-3); drawr(4,7);
 drawr(4,-7); drawr(9,3);
 drawr(-6,-9); drawr(6,-18);
 drawr(-26,0);
end;

procedure napis;
var nap:string[12]; j:integer;
begin
 nap:='DEMONSTRACJA';
 grafpen(2); tag;
 for j:=1 to 12 do
   begin
     plot(544,390-j*20);
     grafout(nap[j]);
   end;
 tagoff;
end;
procedure ryspunkty;
var lan:string[6]; j:integer;
begin
 grafpen(1);
 str(ilpun:6,lan);
 plot(520,256);
 tag;
 for j:=1 to 6 do
  grafout(lan[j]);
 tagoff;
end;
procedure ryspoziom;
var lan:string[2]; j:integer;
begin
 grafpen(1);
 str(poz:2,lan);
 plot(544,336);
 tag;
 for j:=1 to 2 do
  grafout(lan[j]);
 tagoff;
end;
procedure ryspps;
var nap:string[7]; j:integer;
begin
  nap:='POZIOM ';
  grafpen(2); tag;
  plot(528,368);
  for j:=1 to 6 do
   grafout(nap[j]);
  nap:='ILOSC  ';
  plot(528,288);
  for j:=1 to 5 do
   grafout(nap[j]);
  nap:='PUNKTOW';
  plot(528,272);
  for j:=1 to 7 do
   grafout(nap[j]);
  tagoff;
end;

{########################################}
{   PROCEDURA DEMONSTRACJI               }
{########################################}

procedure
 proboj(i:integer; var w:boolean);
  var j:integer; ilp:integer;
begin
  j:=nr-i; ilp:=0;
  repeat
   j:=j+1; w:=false;
   if j>8 then j:=1;
   ilp:=ilp+1;
   if a[j] and b[i+j] and c[i-j]
     then
       begin
        x[i]:=j;
        a[j]:=false; b[i+j]:=false;
        c[i-j]:=false;
        if i<8
          then
            begin
               proboj(i+1,w);
               if w=false
                 then
                   begin
                     a[j]:=true;
                     b[i+j]:=true;
                     c[i-j]:=true;
                   end;
            end
          else
            w:=true;
       end;
  until w or (ilp=8);
end;
```

```
  {#######################################}
  {   PROCEDURY KOMUNIKATOW               }
  {#######################################}
procedure kom1;
var c:char;
begin
repeat
nr:=0;
gotoxy(1,23);
WRITELN
(' PODAJ NUMER WIERSZA                  ');
sound(135,120,80,50,0,0,0);
repeat until keypressed;
READ(kbd,c);
nr:=ord(c)-48;
until (nr>0) and (nr<9);
gotoxy(1,23);
WRITE
('                                      ');
end;
 procedure kom2;
 var r:char;
 begin
gotoxy(1,23);
WRITELN
(' BLEDNY RUCH - TRACISZ 25 PUNKTOW     ');
WRITE
(' WCISNIJ DOWOLNY KLAWISZ              ');
repeat
        sound(2,100,13,20,0,0,0);
        sound(1,150,20,10,0,0,0);
until  keypressed;
READ(kbd,r);
gotoxy(1,23);
WRITELN
('                                      ');
WRITE
('                                      ');
end;
procedure kom3;
var r:char;
begin
gotoxy(1,23);
WRITELN
('NIEMOZLIWE USTAWIENIE-TRACISZ 200 PUNKT.');
WRITE
(' WCISNIJ DOWOLNY KLAWISZ              ');
repeat
 sound(1,random(150)+20,10,10,0,0,0);
 sound(1,random(100)+50,10,10,0,0,0);
until keypressed;
READ(kbd,r);
gotoxy(1,23);
WRITELN
('                                      ');
WRITE
('                                      ');
end;
procedure kom6;
var r:char;
begin
gotoxy(1,23);
WRITELN
(' TO JUZ BYLO  - TRACISZ 200 PUNKTOW  ');
WRITE
(' WCISNIJ DOWOLNY KLAWISZ              ');
repeat
 sound(1,random(40)+60,5,5,0,0,0);
 sound(2,random(10)+80,5,5,0,0,0);
until keypressed;
READ(kbd,r);
gotoxy(1,23);
WRITELN
('                                      ');
WRITE
('                                      ');
end;
procedure pytaj(var y:char);
begin
repeat
  gotoxy(1,24);
  WRITE(' GRASZ DALEJ (t/n) ');
  repeat
   sound(1,50,15,5,0,0,0);
   sound(2,random(10)+80,15,5,0,0,0);
  until keypressed;
  READ(kbd,y);
until (y='t') or (y='T') or (y='n')
      or (y='N');
gotoxy(1,23);
WRITELN
('                                      ');
WRITE('                    ');
end;
procedure koms;
var j:integer;
begin
for j:=1 to 5 do
begin
gotoxy(1,23);
WRITE
(' DLACZEGO SIE POWTARZASZ              ');
sound(1,20,5,5,0,0,0);
sound(2,40,5,5,0,0,0);
end;
end;
{#######################################}
{   PROCEDURY INICJACJI DANYCH          }
{#######################################}
procedure warp;
begin
 for i:=1 to 8 do
  begin
  a[i]:=true;
  x[i]:=0;
  end;
 for i:=2 to 16 do
  b[i]:=true;
 for i:=-7 to 7 do
  c[i]:=true;
end;
procedure inicdanych;
begin
 grafwindow(0,60,640,400);
 poz:=1; ilpun:=0;
 for m:=1 to 92 do
 for n:=1 to 8 do
  tuw[m,n]:=0;
 warp;
 ink(0,26,26);
 ink(1,9,9);
 ink(2,0,0);
 ink(3,6,6);
 paper(2); pen(0);
 grafpaper(0); border(9,9);
 co:=' ';
 y:=' ';
 w:=false;
end;
procedure czyszcz;
begin
clg(0);
for i:=25 downto 22 do
begin
gotoxy(1,i);
WRITE
('                                      ');
end;
end;
{#######################################}
{ PROCEDURA SPRAWDZAJACA USTAWIENIE     }
{#######################################}
procedure
probojsam(ko:integer; var w:boolean);
var j:integer; ilp:integer;
    tnum:array [1..8] of boolean;
begin
 ilp:=ko-1; w:=false;
 for j:=1 to 8 do
  tnum[j]:=false;
 for j:=1 to 8 do
  if x[j]>0 then
    tnum[x[j]]:=true;
 while ((ilp<8) and (w=false)) do
  begin
  repeat
  kom1;
  if tnum[nr]=false then
   ilp:=ilp+1
  else koms;
  until (tnum[nr]=false);
  tnum[nr]:=true;
  j:=nr;
  if a[j] and b[ko+j] and c[ko-j]
    then
      begin
       x[ko]:=j;
       a[j]:=false; b[ko+j]:=false;
       c[ko-j]:=false;
       ryshet(ko);
       ilpun:=ilpun+25;
       ryspunkty;
       w:=true;
      end
    else
      begin
       ilpun:=ilpun-25;
       ryspunkty;
       kom2;
      end;
 end;
 if (w=false)
   then
     begin
       ilpun:=ilpun-200;
       ryspunkty;  kom3;
     end;
end;

{#######################################}
{    MODUL STERUJACY PROCEDURY GRA      }
{#######################################}
begin
mode(1);
inicdanych;
czyszcz;
repeat
repeat
gotoxy(1,23);
WRITE
('DEMONSTRACJA-"d",GRA-"g",KONIEC-"k" ');
repeat
sound(1,20+random(150),15,15,10,0,0);
sound(2,30+random(200),12,12,20,0,0);
until keypressed;
READ(kbd,co);
until (co='d') or  (co='g') or (co='k')
   or (co='D') or  (co='G') or (co='K');
gotoxy(1,23);
WRITE
('                                      ');
if   (co='d') or  (co='g') or
     (co='D') or  (co='G')
then
begin
if (co='d') or (co='D')
then
begin
nr:=RANDOM(7)+1;
warp;
proboj(1,w);
clg(0);
napis;
rysszach;
for i:=1 to 8 do
ryshet(i);
gotoxy(2,23);
WRITE('WCISNIJ DOWOLNY KLAWISZ');
repeat
sound(1,90,15,15,10,0,0);
sound(2,30,12,12,20,0,0);
until keypressed;
READ(kbd,y);
gotoxy(2,23);
WRITE('                        ');
end
else
begin
czyszcz;    clg(0);    ryspps;
rysszach;
ryspunkty;
ryspoziom;
repeat
y:=' '; warp;
i:=1;
repeat
probojsam(i,w);
if (w=false)
then
begin
pytaj(y);
if (y='t') or (y='T')
then
begin
clg(0); ryspps; ryspunkty; ryspoziom;
rysszach; warp;
i:=1;
end;
end
else
i:=i+1;
until (y='n') or (i=9) or (y='N');
if i=9  then
begin
m:=1;
repeat
n:=1;
w:=false;
repeat
if tuw[m,n]<>x[n]
then
w:=true
else
n:=n+1;
until w or (n=9);
if w  then m:=m+1 else m:=93;
until (m=93);
if not  w
then
begin
ilpun:=ilpun-200;
ryspunkty;
kom6;
pytaj(y);
if (y='t') or(y='T') then
begin
clg(0); ryspps; ryspunkty;
ryspoziom; rysszach;
end;
end
else
begin
for n:=1 to 8 do
tuw[poz,n]:=x[n];
gotoxy(2,23);
WRITE
('BRAWO-MOZESZ PRZEJSC NA KOLEJNY POZIOM');
sound(1,10,1,5,10,0,0);
sound(2,40,2,2,20,0,0);
pytaj(y);
if (y='t') or (y='T') then
begin
clg(0); ryspps; ryspunkty;
rysszach;
poz:=poz+1;
ryspoziom;
end;
end;
end;
until (poz>93) or (y='n') or (y='N');
if (y='n') or (y='N')  then co:='k'
else
begin
gotoxy(1,23);
WRITE('  JESTES MISTRZEM  !!!');
sound(1,120,90,5,10,0,0);
sound(2,40,90,20,20,0,0);
co:='k';
end;
end;
end;
until (co='k') or (co='K');
end;
```

Przygotowanie drukarki GEMINI-10Xi do drukowania wersji źródłowych programów

Zestaw mikrokomputerowy AMSTRAD CPC 6128 z drukarką GEMINI—10Xi może być wykorzystywany do drukowania wersji źródłowych programów. Dokonuje się tego za pomocą komendy LIST#:8. Jednak uzyskany w ten sposób wydruk ma następujące mankamenty:

— brak stronicowania;
— pisanie długich instrukcji na perforacji i poza papierem;
— brak możliwości uzyskania innych formatów niż A4.

Problem ten można rozwiązać stosując kody sterujące pracą drukarki.

Prezentowany program umożliwia sformatowanie wydruków poprzez wstawienie odpowiednich kodów, zwalniając użytkownika od konieczności ich znajomości. Rola użytkownika ogranicza się do wyboru następujących opcji (w nawiasie wartości standardowe):

— długość strony (12 cali);
— rodzaj czcionki (10 znaków/cal);
— margines górny (2 linie);
— margines dolny (2 linie);
— margines lewy (2 znaki);
— szerokość tekstu (w zależności od rodzaju czcionki: 68, 83, 118 znaków);
— odstęp między wierszami (9/72 cala);
— druk wytłuszczony ? (T/N)

U dołu ekranu jest wyświetlana informacja, że naciśnięcie klawisza ENTER powoduje przyjęcie wartości podanej w nawiasach. Podawane przez użytkownika wartości poddawane są wnikliwej kontroli formalnej i merytorycznej.

Kontrola formalna polega na zbadaniu, czy wprowadzana wartość jest typu numerycznego (podprogram 2000—2030). Kontrola merytoryczna polega na sprawdzeniu każdego z podanych parametrów z wartościami granicznymi charakterystycznymi dla odpowiedniego kodu sterującego. W przypadku podania błędnego parametru na ekranie jest sygnalizowany błąd i należy powtórnie podać żądaną wartość.

Ustawienie drukarki (podanie odpowiednich kodów sterujących) odbywa się w instrukcji 500.

Po podaniu wszystkich parametrów należy załadować do pamięci program, który ma być drukowany oraz wprowadzić instrukcję LIST#8.

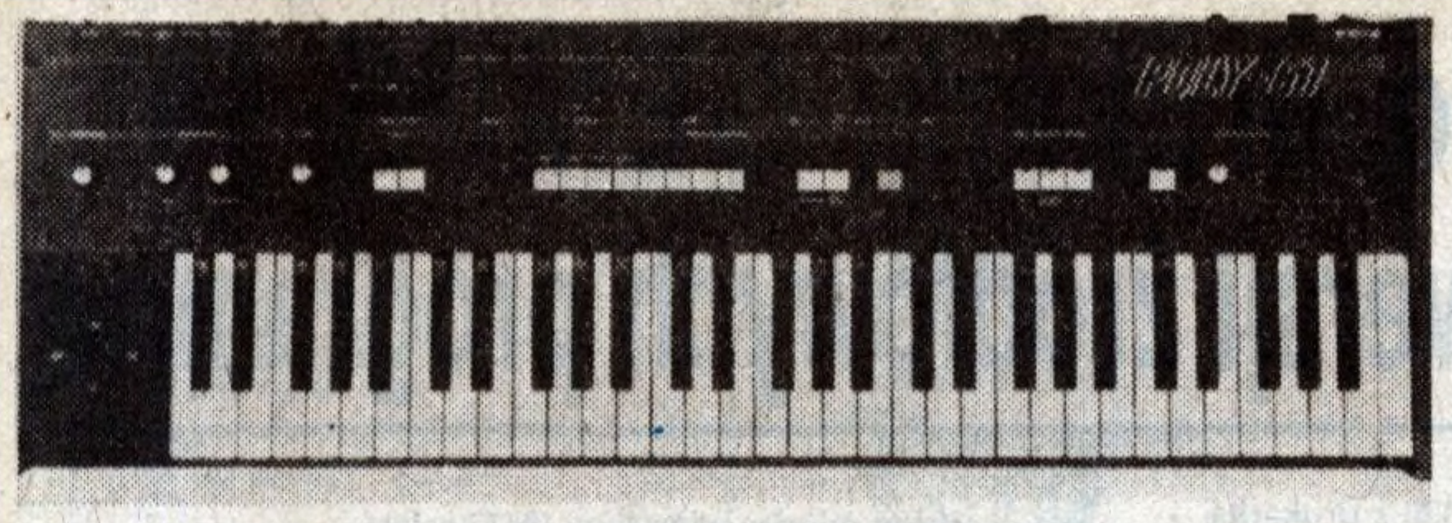
Poszukiwanie różnorodnych form „listingów" satysfakcjonujących Czytelników pozostawiamy posiadaczom drukarek GEMINI—10Xi.

Mieczysław SKONIECZNY
Robert ZUGCHOR

```
10 MODE 2:INK 0,1:INK 1,26:PAPER 0:PEN 1:BORDER 1
20 PRINT"PRZYGOTOWANIE DRUKARKI  gemini-10Xi  DO DRUKOWANIA WERSJI ZRODLOWYCH PR
OGRAMOW"
25 WINDOW £1,1,80,22,25:PAPER £1,1:PEN £1,0:CLS £1:LOCATE £1,1,2:PRINT £1," U W
A G A :  Nacisniecie klawisza ENTER powoduje przyjec
ie wartosci podanych                  w nawiasach."
30 LOCATE 1,3:INPUT "Dlugosc strony (12 cali):
                ",a$
40 IF a$="" THEN a$="12"
45 a=3:GOSUB 2000
46 IF blad=1 THEN GOTO 30
50 dlstr=VAL(a$)
60 IF dlstr<1 OR dlstr>32 THEN GOSUB 1000:GOTO 30
70 LOCATE 1,5:PRINT"  Inne warianty : 12 znakow/cal  lub 17 znakow/cal"
80 LOCATE 1,4:INPUT "Rodzaj czcionki (10 znakow/cal):
                ",a$
90 IF a$="" THEN a$="10"
95 a=4:GOSUB 2000
96 IF blad=1 THEN GOTO 80
100 rp=VAL(a$)
110 IF rp<>10 AND rp<>12 AND rp<>17 THEN GOSUB 1000:GOTO 80
120 IF rp=10 THEN rp=80
130 IF rp=12 THEN rp=77
140 IF rp=17 THEN rp=15
200 LOCATE 1,6:PRINT"  Maksymalny margines - polowa strony"
210 LOCATE 1,5:INPUT "Margines gorny (2 linie):
                ",a$
220 IF a$="" THEN a$="2"
225 a=5:GOSUB 2000
226 IF blad=1 THEN GOTO 210
227 mg=VAL(a$)
230 IF mg<1 OR mg>INT(dlstr/2*6) THEN GOSUB 1000:GOTO 210
250 LOCATE 1,7:PRINT"  Maksymalny margines - polowa strony"
260 LOCATE 1,6:INPUT "Margines dolny (2 linie):
                ",a$
270 IF a$="" THEN a$="2"
275 a=6:GOSUB 2000
276 IF blad=1 THEN GOTO 260
277 md=VAL(a$)
280 IF md<1 OR md>INT(dlstr/2*6) THEN GOSUB 1000:GOTO 260
300 LOCATE 1,8:PRINT "  Maksymalny margines - 30 znakow"
310 LOCATE 1,7:INPUT "Margines lewy (2 znaki):
                ",a$
320 IF a$="" THEN a$="2"
325 a=7:GOSUB 2000
326 IF blad =1 THEN GOTO 310
327 ml=VAL(a$)
330 IF ml<1 OR ml>30 THEN GOSUB 1000:GOTO 310
350 x=70-ml
360 IF rp=77 THEN x=85-ml
370 IF rp=15 THEN x=120-ml
380 LOCATE 1,8:PRINT "Szerokosc drukowanego tekstu (";x;" znakow):":LOCATE 76,8:
INPUT "",a$
390 IF a$="" THEN a$=MID$(STR$(x),2,10)
395 a=8:GOSUB 2000
396 IF blad=1 THEN GOTO 380
397 sd=VAL(a$)
400 IF sd<1 OR sd>x THEN GOSUB 1000:GOTO 380
405 LOCATE 1,10:PRINT"  Podaj licznik ulamka  n/72";
410 LOCATE 1,9:INPUT "Odstep miedzy wierszami (9/72 cala):
                ",a$
415 IF a$="" THEN a$="9"
416 a=9:GOSUB 2000
417 IF blad=1 THEN GOTO 410
418 od=VAL(a$)
420 IF od<9 OR od>255 THEN GOSUB 1000:GOTO 410
440 LOCATE 1,10:INPUT "Druk wytluszczony ? ( T / N ):
                ",a$
450 IF a$="" THEN GOTO 440
460 IF UPPER$(a$)<>"T"AND UPPER$(a$)<>"N" THEN a=10:GOSUB 1000:GOTO 440
470 tl=0
480 IF UPPER$(a$)="T" THEN tl=1
500 PRINT£8,CHR$(27) CHR$(67) CHR$(0) CHR$(dlstr);CHR$(27) CHR$(rp);CHR$(27) CHR
$(114) CHR$(mg+1);CHR$(27) CHR$(78) CHR$(md);CHR$(27
) CHR$(108) CHR$(ml);CHR$(27) CHR$(81) CHR$(ml+sd);CHR$(27) CHR$(65) CHR$(od);CH
R$(27) CHR$(50);CHR$(27) CHR$(72-tl);
510 PRINT£8,CHR$(12);
520 CLS £1:PRINT£1,"":PRINT£1,"  1. Zaladuj program, ktory chcesz drukowac komen
da LOAD                          2. Worowadz komende
  LIST £8"
999 END
1000 LOCATE 1,a:PRINT"Niepoprawny parametr
                ";
1010 FOR t=1 TO 1500:NEXT t
1020 RETURN
2000 blad=0
2005 FOR i=1 TO LEN(a$)
2010 IF ASC(MID$(a$,i,1))<48 OR ASC(MID$(a$,i,1))>57 THEN blad =1
2015 NEXT i
2020 IF blad=1 THEN LOCATE 1,a:PRINT"  Wartosc nienumeryczna ";a$;"
                ":FOR t=1 TO 1500:NEXT
2030 RETURN
```

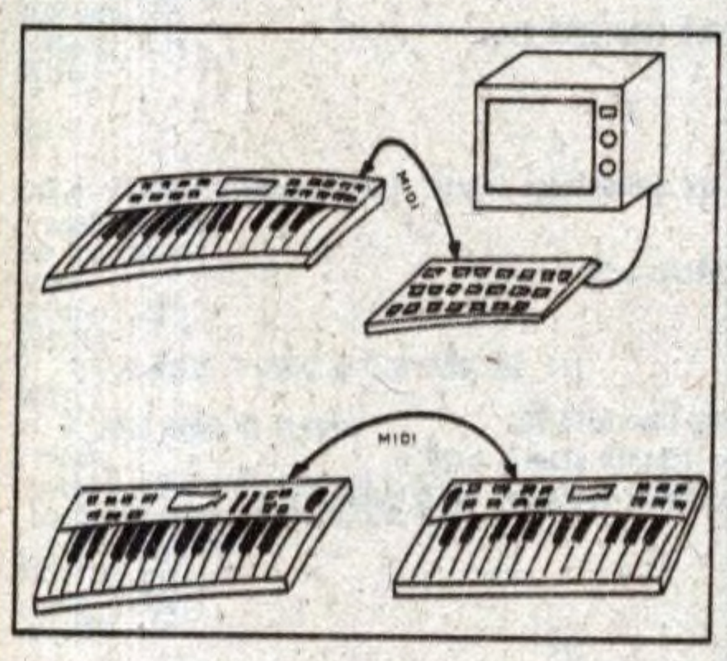
NUTY Z KOMPUTERA (1)

Krzysztof POŹNIAK, Jarosław ZIEMBICKI



Syntezatory muzyczne mają już długą historię. Od najprostszych monofonicznych urządzeń analogowych doszliśmy do skomputeryzowanych układów mogących naśladować jednocześnie całą gamę instrumentów i wytwarzać niezwykłe, nigdzie dotąd nie uzyskiwane dźwięki. Bogata paleta możliwości została okupiona koniecznością programowania licznych parametrów dźwięku. Obecnie produkowane syntezatory są to wyspecjalizowane programowane układy cyfrowe, gdyż tylko takie urządzenia zapewniają wysoką jakość brzmienia instrumentu. Niestety każdy kij ma dwa końce. Programowanie brzmień jest trudną i pracochłonną czynnością. Aby oszczędzić muzykom wiele godzin ciężkiej pracy producenci dołączają do swoich produktów tzw. *cartidge*, **czyli pakiety już zaprogramowanych brzmień. Innym, bardziej ambitnym sposobem jest programowanie za pomocą komputera. Specjalne programy umożliwiają graficzne tworzenie dźwięków na ekranie monitora, wygodną edycję parametrów itp. Komputer przetwarza pomysły muzyka na odpowiedni ciąg danych i przysyła je do syntezatora. W tym miejscu powstaje bardzo ważny problem: w jaki sposób komputer ma nawiązać kontakt z syntezatorem?**

Aby zapewnić pełną łączność pomiędzy wszystkimi instrumentami muzycznymi i dowolnymi komputerami przyjęto specjalny rodzaj transmisji nazywany MIDI (z angielskiego: *Musical Instrument Digital Interface*). Urządzenia wyposażone w system MIDI



Rys. 1

można ze sobą łączyć i przesyłać między nimi informacje. Dotyczy to zarówno transmisji między komputerem i urządzeniem muzycznym, jak również między samymi urządzeniami muzycznymi (rys. 1). Standard MIDI umożliwia łączność z 16 niezależnymi, wspólnie połączonymi urządzeniami, potencjalne możliwości współpracy są zatem olbrzymie. Poprzez MIDI jesteśmy w stanie programować syntezatory, perkusje elektryczne, wzmacniacze estradowe, efekty elektroniczne (np. delay) itp.

Każdy, kto choć trochę interesował się muzyką elektroniczną zauważył, że muzycy posługują się całymi zestawami instrumentów, które znakomicie ze sobą i z muzykiem współpracują. To zgranie jest możliwe dzięki specjalnemu urządzeniu: **sekwencerowi**. Sekwencer w określonym (uprzednio zaprogramowanym) tempie wysyła do połączonych z nim instrumentów rozkazy generacji dźwięków i ustala odpowiednie brzmienia. Z naszego punktu widzenia wykonuje zapisany przez muzyka program.

Do chwili obecnej, na łamach miesięczników związanych z komputerami, pojawiły się artykuły opisujące parametry poszczególnych dźwięków. Bardzo ciekawym i polecanym przez nas jako wstęp jest ciąg artykułów Jacka Jędrzejowskiego o skomputeryzowanej muzyce, jaki ukazał się w Informiku" (dodatek do „Młodego Technika").

W naszym artykule chcemy pokazać muzykom amatorom, jak można programować utwory za pomocą komputera. Tak napisane utwory będą odtwarzane, bądź za pomocą wbudowanych w komputer generatorów dźwięku, bądź przez instrumenty elektroniczne (np. syntezatory i perkusje).

Pragniemy w tym miejscu wtrącić bardzo ważną uwagę: każdy komputer nadaje się do transmisji MIDI i może służyć jako sekwencer dla instrumentów elektronicznych. Z powodzeniem wypróbowaliśmy komputery ZX Spectrum i Sinclair QL, jako sterowniki syntezatorów CASIO CZ 1000, KORG POLY 800, KORG DW 6000 YAMAHA DX 7, ROLAND & YUNO 1 i perkusji elektronicznych ROLAND TR 505, YAMAHA RX 15.

Tworzenie muzyki za pomocą komputera upodabnia się do pisania programu w określonym języku wyższego rzędu (np. Basicu, Pascalu itp.) i dlatego dla wyjaśnienia kilku ważnych pojęć, którymi będziemy się posługiwać, wykorzystamy następujące porównanie:

STRUKTURA PROGRAMU

1) każdy program składa się z rozkazów,
2) rozkazy w programie są tak ułożone, że komputer dokładnie wie, w jakiej kolejności ma je wykonać,
3) dobry program ma budowę strukturalną i składa się z podprogramów (cegiełek), które połączone w odpowiedni sposób budują program.

Jeżeli przeprowadzimy następujące powiązania/znaczeń:

a) dźwięk — rozkaz,
b) takt — podprogram,
c) utwór — program,

to otrzymamy:

STRUKTURA UTWORU

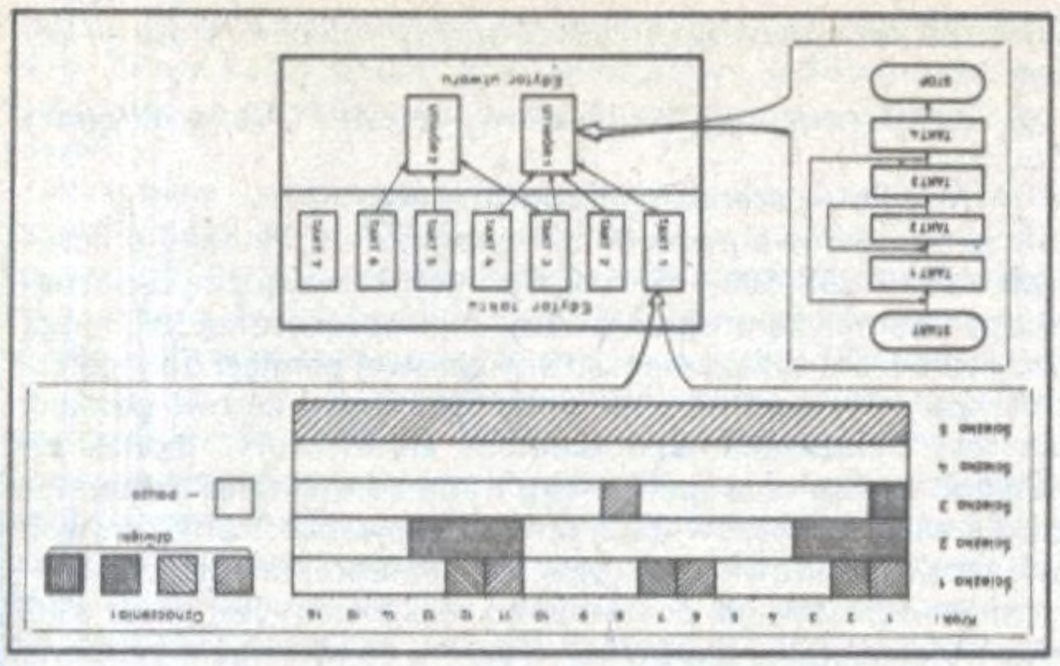
1) każdy utwór składa się z dźwięków,
2) dźwięki są wykonywane w ściśle określonej kolejności,
3) utwór ma budowę strukturalną i składa się z taktów.

Każdy programista wie, jak ważnym narzędziem w jego pracy jest edytor. Za jego pomocą tworzymy, poprawiamy i rozbudowujemy program.

Nasz sekwencer posiada dwa niezależne edytory: edytor taktu (pozwala na budowanie i korekcję taktów) i edytor utworu (za jego pomocą z istniejących taktów składamy utwór).

Podstawową strukturą sekwencera przedstawia rysunek 2.

a) stronę edycyjną obsługują wyżej opisane edytory,
b) nadrzędną strukturą programową jest utwór (jednocześnie zapisanych utworów



Rys. 2

może być kilka). Każdy utwór zbudowany jest z dowolnej ilości taktów, które mogą się wzajemnie zapętlać (powtarzać) określoną ilość razy. Zapętlenia mogą wnikać w głąb siebie, analogicznie, jak pętle FOR—NEXT w programie komputerowym.

c) podrzędną strukturą programową jest takt. Jest to nierozdzielna sekwencja dźwięków. Takt określa kilka parametrów:

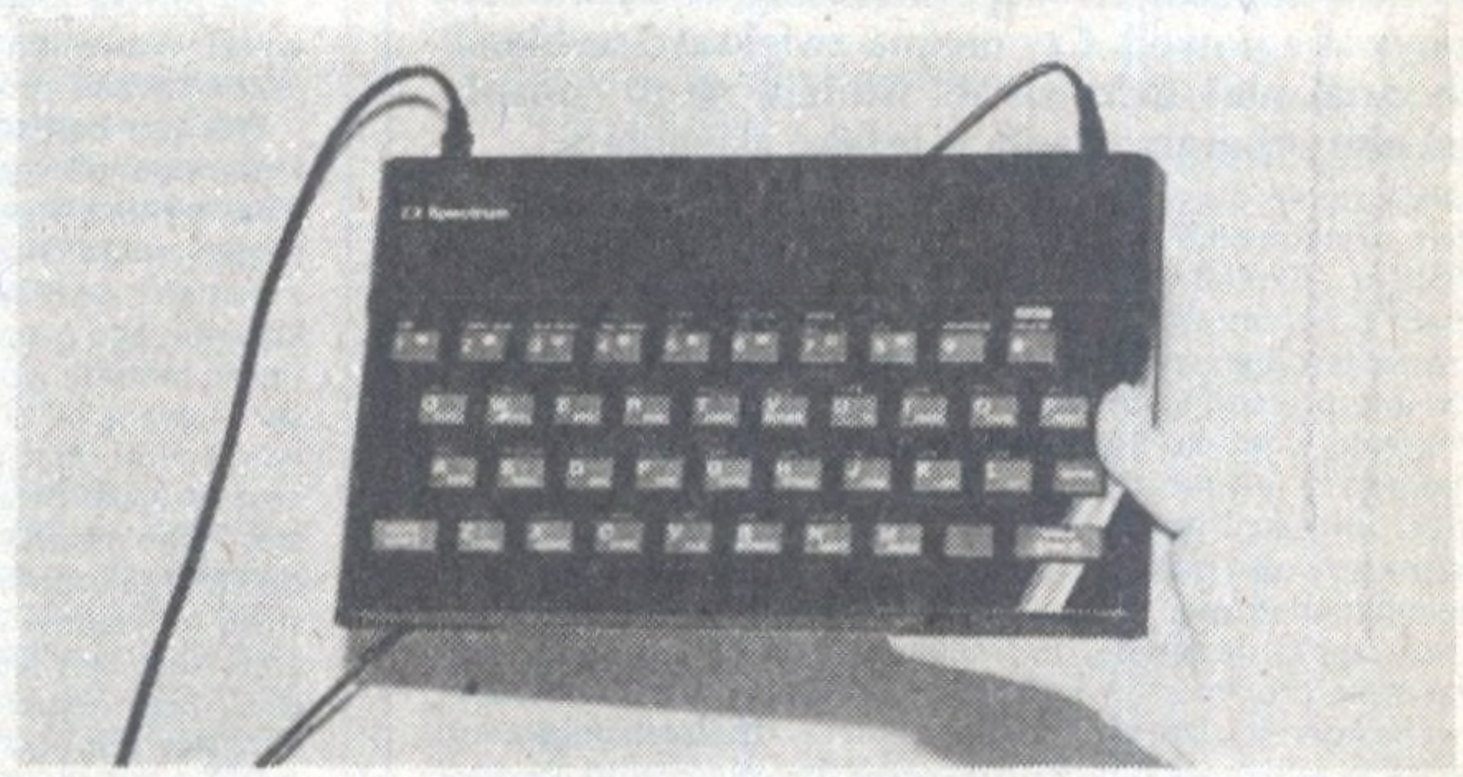
parametr 1: ilość kroków — w pewien sposób łączy się z metrum taktu i informuje ile maksymalnie dźwięków może zmienić się na jednym śladzie. Na rys. 2 przedstawiony został przykładowy takt posiadający 16 kroków. Oznacza to, że na jednym śladzie można umieścić 16 kolejno zmieniających się dźwięków (nazywanych szesnastkami), bądź 8 dwukrotnie dłuższych dźwięków (ósemek), albo dźwięków o innych długościach przedzielonych przerwami (pauzami). Może wystąpić ścieżka pusta (przykładowo ścieżka 4) lub ścieżka generująca dźwięk ciągły (tak jak ścieżka 5).

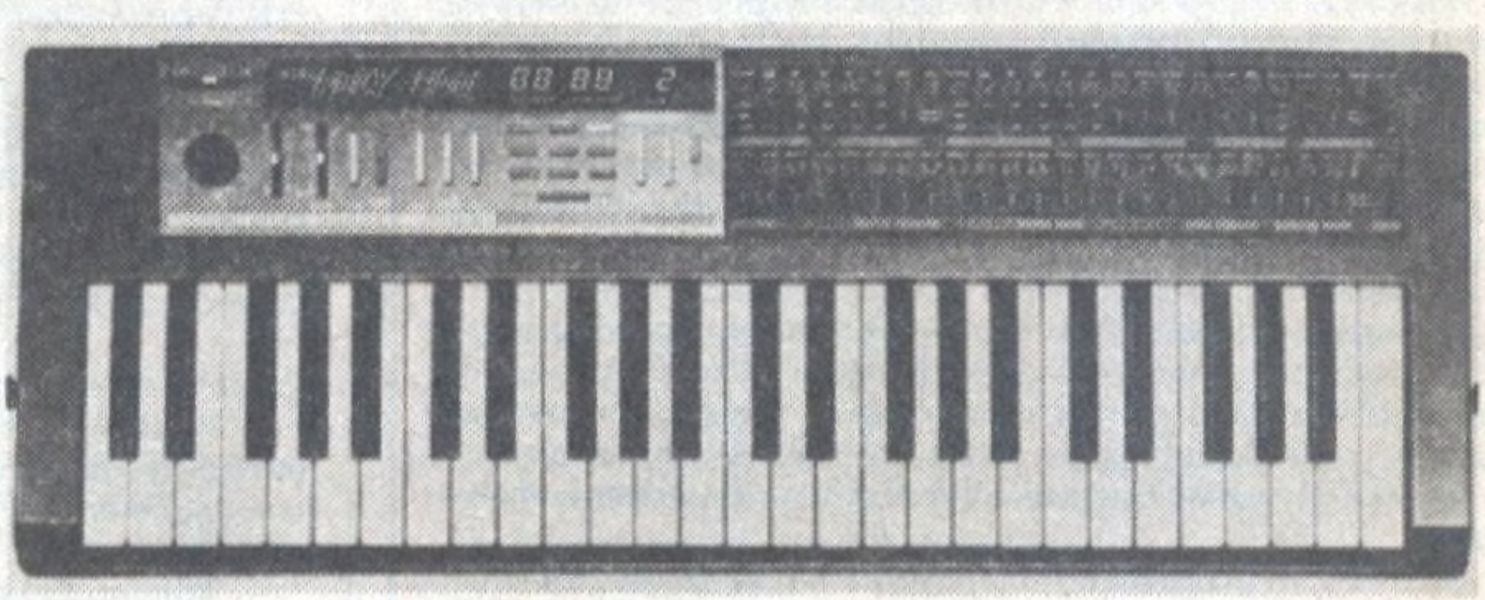
parametr 2: tempo — informuje jak szybko mają następować po sobie kolejne kroki. Każdy takt może posiadać inne tempo.

parametr 3: ilość ścieżek — każda ścieżka zawiera informacje przekazywane jednemu źródłu dźwięku (może być to generator w syntezatorze, lub pojedynczy instrument w perkusji elektronicznej). Ilość ścieżek zależy od liczby niezależnych źródeł dźwięku wykorzystywanych w danym takcie. Ponieważ sekwencer przystosowany jest do sterowania kilkoma instrumentami jednocześnie, stąd ścieżki można w dowolny sposób przypisać poszczególnym instrumentom (np. ścieżki 1—4 mogą obsługiwać syntezator, ścieżki 5—8 generatory komputera, a pozostałe perkusję).

Pragniemy na zakończenie pierwszej części artykułu wyjaśnić, że prezentowany sekwencer ma dosyć ubogą strukturę i należy go analizować bardziej od strony dydaktycznej niż eksploatacyjnej. Stanowi on jednak dobrą bazę do rozbudowy przez poszczególnych użytkowników. Jego prostota zapewnia przejrzystość działania, a nam umożliwia przedstawienie podstawowych problemów programowania utworów i ich transmisji w systemie MIDI.

Zwracamy się w tym miejscu do czytelników zainteresowanych opisanymi problemami o kontakt z redakcją. „IKS" może stać się czasopismem, na łamach którego, tematyka związana z programowaniem muzyki za pomocą komputera znajdzie godne miejsce. Warto tę szansę wykorzystać!





(Foto: Jan Zelman)

Programowanie mikroprocesora 6502

Henryk KRASUSKI

Każdy użytkownik mikrokomputera ATARI XL/-XE często irytuje się, obserwując realizację programów napisanych w języku BASIC — są one wykonywane powoli. Czy można zwiększyć szybkość ich działania? Oczywiście, istnieje na to sposób. Jest nim programowanie w języku asemblera.

Na wstępie pragnę uczynić jedno zastrzeżenie: nie namawiam użytkowników ATARI do zaniechania programowania w BASIC-u. ATARI BASIC jest dobrym i bardzo wygodnym narzędziem programowania, a to, że niektóre operacje wykonuje powoli, jest jego wadą, charakterystyczną dla wszystkich języków interpretacyjnych, którą łatwo wyeliminować, włączając do programu w języku BASIC procedury w kodzie maszynowym. Operacja ta jest prosta, nawet dla początkującego programisty dzięki instrukcji języka BASIC X=USR (...), za pomocą której można wywołać podprogram maszynowy i przekazać do niego wiele parametrów.

Jest rzeczą oczywistą, że trzeba wiedzieć, w jakiej sytuacji włączenie procedury w kodzie maszynowym przyniesie oczekiwane korzyści, tzn. wydatnie przyspieszy działanie programu, a w jakiej zabieg ten okaże się nieopłacalny. Na pewno nie warto programować operacji arytmetycznych, czy arytmetyki zmiennoprzecinkowej w kodzie maszynowym. Istnieją natomiast obszary, w których zastosowanie kodu maszynowego zwiększa szybkość działania programu od kilku do kilkunastu razy, a w skrajnych przypadkach nawet 1000 razy. Obszarami tymi są:

— generowanie dźwięków;
— wyprowadzanie dużych zbiorów informacji na ekran;
— pętle programowe w języku BASIC, realizujące przesłania pomiędzy bajtami pamięci, zmieniające zawartość obszarów pamięci oraz wyszukujące z nich bajty o określonej zawartości;
— wykonywanie operacji logicznych na pojedynczych bitach.

W ostatnim przypadku mniej istotna jest szybkość działania, lecz fakt, że operacje tego typu są niemożliwe do wykonania z poziomu języka BASIC.

Tworzenie programów w kodzie maszynowym wymaga znajomości mikroprocesora MOSTEC 6502, jego architektury, listy rozkazów i sposobu ich wykonywania oraz podstawowych procedur systemu operacyjnego. Ponadto przydatny jest do tego asembler (nie jest niezbędny), tj. program tłumaczący rozkazy z postaci mnemonicznej na ich kod maszynowy. Godnym polecenia jest pakiet ATARI Assembler Editor, zawierający oprócz asemblera, edytor — umożliwiający pisanie i poprawianie programów oraz debugger — program przeznaczony do testowania napisanych programów maszynowych. Zaletą tego pakietu jest możliwość współpracy z magnetofonem i stacją dysków, co z punktu widzenia wielu użytkowników ATARI nie posiadających stacji dysków jest rzeczą o istotnym znaczeniu.

Na początek kilka słów o architekturze mikroprocesora 6502. Do jej elementów należą:

a) 8-bitowa szyna danych;
b) 16-bitowa szyna adresowa;
c) jeden akumulator — rejestr A;
d) dwa 8-bitowe rejestry indeksowe — rejestr X i Y;

Rejestry te można również wykorzystywać, podobnie jak akumulator, tzn. można przechowywać w nich dane, przesyłać za ich pośrednictwem dane pomiędzy bajtami pamięci, itp.;

e) rejestr stanu mikroprocesora, zwany również rejestrem flagowym lub rejestrem F;

f) stos mikroprocesora. Stanowi go pierwsza strona pamięci mikrokomputera, tj. bajty o adresach od 256 do 511 ($100 — $1FF). Ze stosem związany jest 9-bitowy wskaźnik stosu — rejestr S;

g) licznik programu — 16-bitowy rejestr PC (ang. Program Counter);

h) dwie linie przerwań mikroprocesora;

i) strona zerowa pamięci mikrokomputera, tj. bajty o adresach od 9 do 255 ($00 — $FF). Czytelnicy, którzy już są zapoznani z problematyką programowania mikroprocesora 6502, mogą być zaskoczeni zaliczeniem strony zerowej pamięci do jego architektury. W literaturze na temat 6502 strona zerowa pamięci nie jest traktowana jako element architektury. Jednak ze względu na istnienie specjalnego trybu adresowania dostępnego dla wielu rozkazów 6502, tzw. adresowania strony zerowej, uważam, że stanowi ona istotny jej element. Ponadto cały mechanizm adresowania pośredniego realizowany jest przez 6502 przy wykorzystaniu bajtów tej strony — co również przemawia za traktowaniem jej jako elementu architektury.

Na bliższe przedstawienie zasługują trzy elementy spośród wyżej wymienionych, a mianowicie: stos i wskaźnik stosu, rejestr F i licznik programu.

Stos jest bardzo ważnym elementem każdego systemu mikrokomputerowego. Można go sobie wyobrazić jako zbiór rejestrów wykorzystywanych przez mikroprocesor przy obsłudze przerwań, do chwilowego przechowywania danych pobranych z pamięci, przy wykonywaniu procedur wywoływanych z wykonywanego programu, itp. Mikroprocesor 6502 stosuje specyficzną metodę wprowadzania i odczytywania informacji ze stosu — zwaną LIFO (ang. last-in, first-out). Metoda ta polega na tym, że informacja zapisana do stosu jako pierwsza, odczytana zostanie jako ostatnia, natomiast informacja wprowadzona do stosu jako ostatnia, będzie pierwszą przy odczycie. Adres stosu, do którego zapisano ostatni element, zawarty jest we wskaźniku stosu (dokładnie jest to ten adres zwiększony o jeden). Wskaźnik stosu jest jedynym 9-bitowym rejestrem mikroprocesora 6502. Jego najbardziej znaczący bit jest zawsze równy jeden. Gdy stos jest pusty, wskaźnik stosu wskazuje adres 511 ($1FF). Przy wprowadzeniu kolejnego bajtu do stosu wskaźnik stosu jest zmniejszony o jeden aż do wartości 256 ($100). Próba wprowadzenia do stosu informacji przy wskaźniku równym 256 prowadzi do przepełnienia stosu i w sytuacji, gdy wartość wskaźnika nie jest kontrolowana, konsekwencją tego może być załamanie systemu. Wywoływanie podprogramów maszynowych z języka BASIC realizowane jest z wykorzystaniem stosu mikroprocesora. Instrukcja:

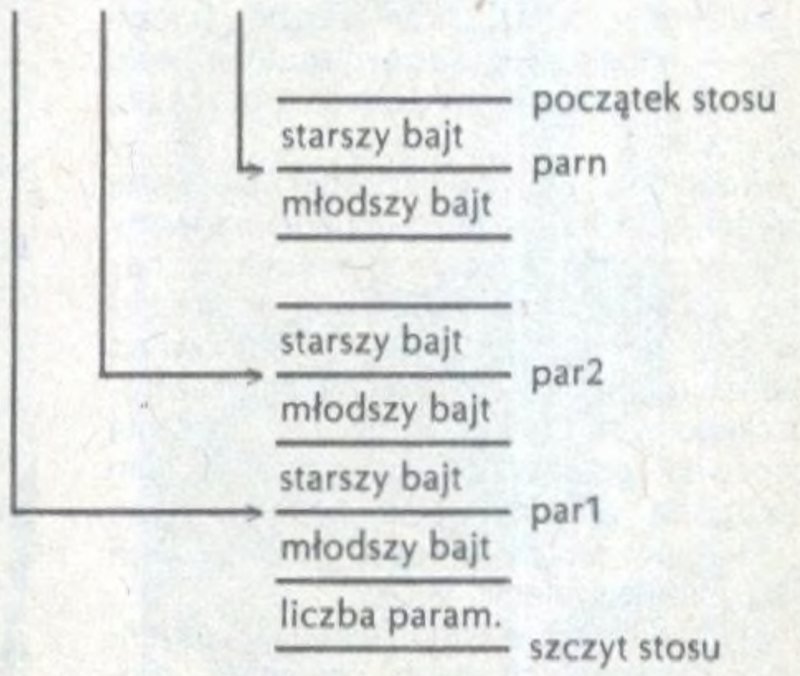
X=USR(adres,par1, par2,...,parn)

działa w sposób następujący:

— na stos przesyłane są dwubajtowe parametry w kolejności odwrotnej od tej, w jakiej zostały wymienione w instrukcji USR, tzn. najpierw wysyłany jest parn i wskaźnik stosu jest zmniejszany o 2;
— kolejno przesyłane są pozostałe parametry... par2, par1. Każdy z nich zajmuje dwa bajty stosu;
— na końcu na stos wysyłana jest jednobajtowa liczba przesłanych parametrów.

Ilustruje to poniższy schemat:

X=USR(adres,par1,par2,...,parn)



Wywołany podprogram maszynowy, przed rozpoczęciem realizacji przypisanej mu funkcji, musi pobrać ze stosu przekazane parametry. Wykonuje to sekwencją rozkazów PLA i STA (patrz lista rozkazów). Nawet w sytuacji, gdy do podprogramu maszynowego nie są przekazywane żadne parametry musi on

usunąć ze stosu bajt zawierający ich liczbę — w tym przypadku zero. Zatem każdy podprogram maszynowy, przeznaczony do współpracy z językiem BASIC musi wykonać rozkaz PLA jako pierwszy.

Wykonany podprogram maszynowy może przekazywać z powrotem do języka BASIC wynik zrealizowanej funkcji. Może to czynić w dwojaki sposób. Jeżeli wynik zrealizowanej funkcji mieści się na dwóch bajtach, to podprogram maszynowy może umieścić go w bajtach strony zerowej o adresach $D4 (młodszy) i $D5 (starszy). Przekazany w ten sposób wynik jest dostępny z poziomu języka BASIC w zmiennej X użytej przy wywołaniu podprogramu maszynowego. Natomiast, jeżeli wynik działania podprogramu maszynowego nie da się przekazać w dwóch bajtach pamięci (np. posortowana tablica stałych tekstowych), to może on być przekazany do języka BASIC w zarezerwowanym dla tego celu obszarze pamięci, którego adres powinien być przekazany do podprogramu maszynowego jako jeden z parametrów.

Rejestr stanu mikroprocesora zawiera szereg wskaźników, które mogą być wykorzystywane przez program maszynowy. Zawartość niektórych z nich może być zmieniana przez programistę dla specyficznych celów, jednak z reguły ich zawartość jest modyfikowana przez mikroprocesor, aby poinformować programistę o aktualnym stanie procesu przetwarzania. Rejestr ten można zobrazować poniższym schematem:

INIVI IBIDIIIZICI rejestr F

Znaczenie poszczególnych wskaźników jest następujące:

N — wskaźnik znaku. Jest ustawiony (N=1), gdy po wykonaniu ostatniego rozkazu w bajcie zawierającym wynik, najbardziej znaczący bit jest równy jeden. Gdy po wykonaniu ostatniego rozkazu najbardziej znaczący bit wyniku jest zerem, wskaźnik N jest zgaszony (N=0). Wskaźnik ten jest ustawiany również przez procedury systemu operacyjnego, obsługujące urządzenia zewnętrzne w przypadku niepomyślnego zakończenia przesłania danych z/do tych urządzeń.

V — wskaźnik przepełnienia. Jest ustawiany w sytuacji, gdy wynik ostatniego rozkazu nie mieści się na siedmiu bitach i do jego przedstawienia musi być użyty najbardziej znaczący bit, który normalnie jest bitem znaku.

B — wskaźnik przerwania programowego. Jest ustawiany, gdy miroprocesor wykona rozkaz przerwania BRK (patrz lista rozkazów).

D — wskaźnik dziesiątego trybu pracy mikroprocesora. Jest ustawiany i gaszony przez programistę. Gdy jest ustawiony mikroprocesor traktuje liczby, na których wykonuje operacje arytmetyczne nie jak liczby binarne, a jak liczby w kodzie BCD.

I — wskaźnik przerwań sprzętowych. Programista ustawiając ten wskaźnik blokuje przerwania tego typu.

Z — wskaźnik zera. Ustawiany jest, gdy wynikiem ostatniego wykonanego rozkazu jest zero. Ustawiany jest również, gdy bajt zawierający zero zostanie pobrany z pamięci do akumulatora lub jednego z rejestrów indeksowych.

C — wskaźnik przeniesienia. Jest ustawiany, gdy wynik dodawania lub odejmowania nie mieści się w jednym bajcie. Na przykład: gdy dodamy do siebie dwie liczby 128 i 129, to otrzymamy 257. W takiej sytuacji wskaźnik C zostałby ustawiony, ponieważ liczby 257 nie da się zapisać na ośmiu bitach. Wskaźnik może być ustawiony również przez rozkazy przesuwające w lewo lub w prawo zawartość bajtów (patrz lista rozkazów).

Do efektywnego łączenia podprogramów maszynowych z programem w języku BASIC wystarczy posługiwanie się czterema z wyżej wymienionych wskaźników: N, Z, C i V.

Licznik programu jest 16-bitowym rejestrem, w którym mikroprocesor przechowuje adres następnego rozkazu, który ma być wykonany. Licznik programu zbudowany jest z dwóch rejestrów 8-bitowych PCL i PCH, natomiast efektywny adres rozkazu do wykonania obliczany jest według znanej zależności adres = (PCL) + 256 * (PCH).

Zestaw rozkazów mikroprocesora składa się z 56 rozkazów. W większości rozkazów można wyodrębnić dwa elementy: kod operacji do wykonania (kod rozkazu) i operand. Kod rozkazu jest liczbą z zakresu 0—255 i zawsze zajmuje jeden bajt pamięci. Operand wskazuje argument rozkazu, tj. bajt w pamięci mikrokomputera lub rejestr mikroprocesora, na którym operacja ma być wykonana. Operand może zajmować jeden albo dwa bajty pamięci lub może w ogóle nie występować. Stąd rozkazy mikroprocesora mogą mieć różną długość — od jednego do trzech bajtów.

bajt	n	n+1	n+2	
| kod |				rozkaz jednobajtowy
| kod | |operand|				rozkaz dwubajtowy
| kod | o p e | r a n d |				rozkaz trzybajtowy

Wiele rozkazów 6502 posiada kilka trybów adresowania. Tryb adresowania zawiera informację o sposbie interpretacji operandu i o jego wielkości. W dalszej części artykułu przedstawione zostały wszystkie tryby adresowania rozkazów mikroprocesora 6502. Dla każdego trybu podany został sposób zapisu operandu wymagany przez pakiet ATARI Assembler Editor oraz sposób, w jaki obliczana jest efektywna wartość argumentu rozkazu. W każdym trybie adresowania operand, o ile występuje, może być liczbą szesnastkową (poprzedzoną znakiem $ lub liczbą dziesiętną.

1. Adresowanie natychmiastowe

kod # operand

Operand jednobajtowy. Argumentem rozkazu jest operand. Np.: rozkaz LDA # 20 powoduje, że do akumulatora zostanie wprowadzona liczba 20.

2. Adresowanie absolutne

kod operand

Operand dwubajtowy. Argumentem rozkazu jest zawartość bajtu, którego adres jest równy wartości operandu.

3. Adresowanie absolutne indeksowane

kod operand, X
kod operand, Y

Operand dwubajtowy. Argumentem rozkazu jest zawartość bajtu o adresie powstałym przez dodanie do wartości operandu zawartości jednego z rejestrów indeksowych X lub Y.

4. Adresowanie strony zerowej

kod operand

Operand jednobajtowy. Argumentem rozkazu jest zawartość bajtu ze strony zerowej. Adres tego bajtu jest równy wartości operandu.

5. Adresowanie strony zerowej indeksowane

kod operand, X

Operand jednobajtowy. Argumentem rozkazu jest zawartość bajtu ze strony zerowej o adresie równym wartości operandu zwiększonej o zawartość rejestru indeksowego X. Jak widać, ten tryb adresowania może używać jako rejestru indeksowego wyłącznie rejestru X. Jednakże istnieją dwa rozkazy stanowiące wyjątek od tej zasady. Są to LDX (załaduj do rejestru X) i STX (prześlij zawartość rejestru X). Rozkazy te mogą używać jako rejestru indeksowego, w tym trybie adresowania — rejestru Y.

6. Adresowanie absolutne pośrednie

kod (operand)

Operand dwubajtowy. Istota adresowania pośredniego polega na tym, że operand nie wskazuje bezpośrednio adresu argumentu, jak to miało miejsce w wyżej wymienionych trybach adresowania, lecz bajt w pamięci — wskaźnik, z którego pobierany jest adres argumentu. Jest to sytuacja analogiczna do tej, która ma miejsce przy wykorzystaniu instrukcji GOTO z języka BASIC, zawierającej zamiast numeru linii nazwę zmiennej, do której wcześniej wpisano numer linii. Jedynym rozkazem mikroprocesora 6502, w którym dozwolone jest użycie tego trybu adresowania, jest rozkaz skoku JMP. Na przykład: rozkaz JMP ($1000) nie skacze bezpośrednio do rozkazu zawartego w bajcie o adresie $1000, lecz jest realizowany w sposób następujący:
— pobierana jest zawartość bajtu o adresie równym wartości operandu i bajtu następnego, tj. o adresie o jeden większym, w przykładzie są to bajty o adresach $1000 i $1001;
— zawartość tych dwóch bajtów traktowana jest jako 16-bitowy adres obliczany według zależności: ($1000)+($1001)*256;
— skok następuje do rozkazu zawartego w bajcie o obliczonym adresie.
Wszystkie inne rozkazy, które mogą używać adresowania pośredniego, muszą jednocześnie używać rejestrów indeksowych X lub Y. Istnieją dwa tryby adresowania pośredniego wykorzystujące te rejestry. Są one przedstawione poniżej.

7. Adresowanie indeksowane pośrednie

kod (operand, X)

Operand jednobajtowy. Ten tryb adresowania wymaga użycia rejestru X. Jego zawartość dodawana jest do wartości operandu. Suma wskazuje adres bajtu na stronie zerowej. Wskazywany

bajt i bajt następny zawierają 16-bitowy adres argumentu rozkazu.

8. Adresowanie pośrednie indeksowane

kod (operand),Y

Operand jednobajtowy. Tryb ten wymaga użycia rejestru indeksowego Y. Adres argumentu rozkazu wyznaczany jest podobnie jak w trybie przedstawionym w punkcie 7. Zasadnicza różnica polega na tym, że zawartość rejestru nie jest dodawana do wartości operandu, a do 16-bitowego adresu, pobranego z dwóch bajtów, z których pierwszy wskazywany jest przez operand. Mówiąc inaczej: w adresowaniu indeksowanym pośrednim wartość indeksu dodawana jest do wartości operandu przed wyznaczeniem efektywnego adresu argumentu rozkazu (preindeksacja). Natomiast w adresowaniu pośrednim indeksowanym wartość indeksu dodawana jest po wyznaczeniu adresu (postindeksacja).

9. Adresowanie względne

kod operand

Operand jednobajtowy. Ten tryb adresowania dozwolony jest jedynie dla rozkazów skoków tzw. krótkich. Operand traktowany jest jako liczba dodatnia, gdy jest mniejszy od 128 lub jako liczba ujemna — równa 256 minus jego wartość, gdy jest nie mniejszy od 128. Liczba ta dodawana jest do zawartości licznika programu, wyznaczając adres, do którego zostanie wykonany skok. Ponieważ licznik programu zawsze wskazuje adres następnego do wykonania rozkazu, a rozkazy skoków „krótkich" zajmują dwa bajty pamięci, to za ich pomocą „skoczyć" można maksymalnie o 129 (127+2) bajtów do przodu i 126 (—128+2) bajtów do tyłu od adresu, w którym znajduje się rozkaz skoku. Podprogram maszynowy, w któm wykorzystano wyłącznie rozkazy skoków z tym trybem adresowania jest relokowalny, tzn. będzie działał poprawnie bez względu na obszar pamięci, w którym zostanie umieszczony. Podprogramy tak skonstruowane mogą być umieszczane w stałych tekstowych języka BASIC. Natomiast podprogramy, w których zastosowano rozkazy skoków tzw. długich (JMP i JSR), muszą rezydować w ściśle określonych miejscach pamięci — są programami nierelokowalnymi. Umieszczenie tych programów w nieodpowiednich obszarach pamięci powoduje ich złe działanie. Wyjątek stanowi użycie rozkazów JSR i JMP do wywołania procedur systemu operacyjnego — podprogram maszynowy jest w tym przypadku również relokowalny.

10. Adresowanie akumulatora

kod A

W tym trybie adresowania nie występuje operand. Argumentem rozkazu jest zawartość akumulatora. ATARI Assembler Editor wymaga, aby jako operand wpisywać literę A. W innych asemblerach wymaganie takie może nie istnieć.

11. Adresowanie wewnętrzne

kod

W tym trybie adresowania również nie występuje operand. Argumentem rozkazu z tym trybem adresowania jest z reguły zawartość jednego z rejestrów mikroprocesora lub wskaźnika z rejestru flagowego.

W przedstawionej poniżej liście rozkazów mikroprocesora 6502 opisane zostały wszystkie rozkazy dostępne dla programisty. Zostały one podzielone na kilka grup. Jako kryterium podziału zastosowano stopień podobieństwa funkcji realizowanych przez różne rozkazy. Dla każdego rozkazu, obok jego nazwy mnemonicznej, podano krótki opis realizowanej funkcji, tryby adresowania, z którymi rozkaz może być użyty, kody szesnastkowe i dziesiętne rozkazu, liczbę zajmowanych bajtów pamięci oraz wskaźniki rejestru flagowego, których stan może ulec zmianie po wykonaniu rozkazu.

ROZKAZY ZMIENIAJĄCE W SPOSÓB BEZPOŚREDNI STAN WSKAŹNIKÓW REJESTRU FLAGOWEGO

Rozkazy z tej grupy dają programiście możliwość bezpośredniej zmiany zawartości wskaźników C, D, I oraz V rejestru F. Każdy z nich zajmuje jeden bajt pamięci i może być użyty wyłącznie z wewnętrznym trybem adresowania. Rozkazami tymi są:

CLC — zeruj wskaźnik przeniesienia C;
CLD — zeruj wskaźnik dziesiętnego trybu pracy D;
CLI — zeruj wskaźnik przerwań I;
CLV — zeruj wskaźnik przepełnienia V;
SEC — ustaw wskaźnik przeniesienia;
SED — ustaw wskaźnik dziesiętnego trybu pracy;
SEI — ustaw wskaźnik przerwań.

Kody szesnastkowe i dziesiętne rozkazów:

CLC — $18/24
CLD — $D8/216
CLI — $58/88
CLV — $B8/184
SEC — $38/56
SED — $F8/248
SEI — $78/120

ROZKAZY SKOKÓW „KRÓTKICH" (WZGLĘDNYCH)

Rozkazy te umożliwiają przeniesienie sterowania w programie maszynowym maksymalnie o 129 bajtów do przodu i 126 bajtów do tyłu od ich własnego adresu w zależności od stanu jednego ze wskaźników V, Z, C i N. Każdy z nich zajmuje dwa bajty pamięci i może być użyty tylko ze względnym trybem adresowania. Nie oddziałują one na żaden ze wskaźników rejestru flagowego. Rozkazami tymi są:

BCC — skocz, gdy wskaźnik C=0;
BCS — skocz, gdy wskaźnik C=1;
BVC — skocz, gdy wskaźnik V=0;
BVS — skocz, gdy wskaźnik V=1;
BNE — skocz, gdy wskaźnik Z=0;
BEQ — skocz, gdy wskaźnik Z=1;
BPL — skocz, gdy wskaźnik N=0;
BMI — skocz, gdy wskaźnik N=1;

Kody szesnastkowe i dziesiętne rozkazów:

BCC — $90/144 BCS — $BO/176 BCV — $50/80
BVS — $70/112 BNE — $DO/208 BEQ — $FO/240
BPL — $10/16 BMI — $30/48

ROZKAZY SKOKÓW „DŁUGICH" (BEZWZGLĘDNYCH)

Są dwa rozkazy umożliwiające przeniesienie sterowania w programie maszynowym w dowolne jego miejsce. Rozkazami tymi są: JMP i JSR.

JMP — skocz do adresu określonego przez operand.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
absolutne	JMP 3000	$4C/76	3
pośrednie	JMP (3000)	$6C/108	3

JMP nie oddziałuje na żadne wskaźniki rejestru F. Używając tego rozkazu z pośrednim trybem adresowania, należy zwrócić uwagę, aby określony przez operand wskaźnik zawierający adres skoku nie znajdował się na granicy dwóch stron pamięci. Na przykład: w rozkazie JMP ($OBFF) młodszy bajt wskaźnika jest ostatnim bajtem strony $OB, a starszy bajt pierwszym bajtem strony $OC. W takiej sytuacji adres skoku jest wyznaczany przez mikroprocesor 6502 w sposób błędny. Natomiast, gdy wskaźnik z adresem skoku nie jest położony na granicy stron pamięci — JMP z pośrednim trybem adresowania działa prawidłowo.

JSR — wykonaj podprogram o adresie określonym przez operand.

Licznik programu jest zwiększony o 2 ((PC)=(PC)+2)) i przesyłany jest na stos. Następnie sterowanie przekazywane jest do rozkazu, którego adres określany jest przez operand. Powrót z podprogramu (tj. wykonanie instrukcji następującej po JSR) realizowany jest za pomocą rozkazu RTS, który powinien być ostatnim rozkazem wywołanego podprogramu. Rozkaz JSR nie oddziałuje na żadne wskaźniki rejestru flagowego. Jedyny dostępny dla tego rozkazu tryb adresowania to adresowanie absolutne, np.: JSR $E456. Kod rozkazu: $20/32. JSR zajmuje trzy bajty pamięci.

ROZKAZY PRZESYŁAJĄCE ZAWARTOŚĆ BAJTU Z PAMIĘCI DO REJESTRÓW A, X, Y

Są trzy takie rozkazy:

LDA — załaduj do akumulatora bajt z pamięci;
LDX — załaduj do rejestru X bajt z pamięci;
LDY — załaduj do rejestru Y bajt z pamięci;

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l.zaj. bajtów
LDA			
natychmiastowe	LDA #14	$A9/169	2
absolutne	LDA 3000	$AD/173	3
absolutne indeksowane X	LDA 3000, X	$BD/189	3
absolutne indeksowane Y	LDA 3000,Y	$B9/185	3
strony zerowej	LDA $A9	$A5/165	2
strony zerowej indeks. X	LDA $A9,X	$B5/181	2
indeksowane pośrednie	LDA ($A9,X)	$A1/161	2
pośrednie indeksowane	LDA ($A9),Y	$B1/177	2
LDX			
natychmiastowe	LDX #14	$A2/162	2
absolutne	LDX 3000	$AE/174	2
absolutne indeksowane Y	LDX 3000, Y	$BE/190	3
strony zerowej	LDX $A9	$A6/166	2
strony zerowej indeks. Y	LDX $A9,Y	$B6/182	2
LDY			
natychmiastowe	LDY #14	$AO/160	2
absolutne	LDY 3000	$AC/172	3
absolutne indeksowane X	LDY 3000, X	$BC/188	3
strony zerowej	LDY $A9	$A4/164	2
strony zerowej indeks. X	LDY $A9, X	$B4/180	2

Rozkazy LDA, LDX, LDY oddziałują na wskaźniki N i Z. Mogą ustawić wskaźnik N, gdy ładowany do rejestru bajt ma najbardziej znaczący bit równy 1. Wskaźnik Z zostanie ustawiony, gdy ładowany do rejestru bajt zawiera zero.

ROZKAZY PRZESYŁAJĄCE ZAWARTOŚĆ REJESTRÓW A, X, Y DO PAMIĘCI

Są to rozkazy komplementarne w stosunku do rozkazów z poprzedniej grupy. Również są trzy takie rozkazy.

STA — zapamiętaj zawartość akumulatora w pamięci;
STX — zapamiętaj zawartość rejestru X w pamięci;
STY — zapamiętaj zawartość rejestru Y w pamięci;

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
STA			
absolutne	STA $D300	$8D/141	3
absolutne indeksowane X	STA $D300, X	$9D/157	3
absolutne indeksowane Y	STA $D300, Y	$99/153	3
strony zerowej	STA 240	$85/133	2
strony zerowej indeks. X	STA 240, X	$95/149	2
indeksowane pośrednie	STA (240, X)	$81/129	2
pośrednie indeksowane	LDA (240), Y	$91/145	2
STX			
absolutne	STX $D300	$8E/142	3
strony zerowej	STX $FO	$86/134	2
strony zerowej indeks. Y	STX $FO, Y	$96/150	2
STY			
absolutne	STY 3000	$8C/140	3
strony zerowej	STY 129	$84/132	2
strony zerowej indeks. X	STY 129, X	$94/148	2

Rozkazy STA, STX, STY nie oddziałują na żadne wskaźniki rejestru flagowego.

ROZKAZY PRZESŁAŃ POMIĘDZY REJESTRAMI A, X, Y

Rozkazy te najczęściej służą do chwilowego przechowania zawartości jednego rejestru w innym. Można również przechowywać zawartość rejestrów w pamięci. Jednak to drugie rozwiązanie jest gorsze. Rozkazy przesyłające zawartość rejestrów z/do pamięci wykonywane są w czasie średnio dwukrotnie dłuższym i zajmują więcej pamięci.

TAX — prześlij zawartość akumulatora do rejestru X;
TAY — prześlij zawartość akumulatora do rejestru Y;
TXA — prześlij zawartość rejestru X do akumulatora;
TYA — prześlij zawartość rejestru Y do akumulatora;

Kody rozkazów:

TAX — $AA/170 TAY — $A8/168
TXA — $8A/138 TYA — $98/152

Do tej grupy należałoby również zaliczyć dwa rozkazy umożliwiające programiście zmianę zawartości wskaźnika stosu (rejestru S). Zmiana zawartości tego rejestru nie może być dokonana w sposób bezpośredni, a jedynie poprzez rejestr X. Rozkazami tymi są:

TSX — prześlij zawartość ośmiu mniej znaczących bitów rejestru S do rejestru X;
TXS — prześlij zawartość rejestru X do ośmiu mniej znaczących bitów rejestru S;

Każdy z tych rozkazów zajmuje jeden bajt pamięci i może być użyty wyłącznie z wewnętrznym trybem adresowania. Oddziałują one, za wyjątkiem TXS, na wskaźniki N i Z rejestru flagowego. TXS może zmienić stan wszystkich wskaźników rejestru F.

TSX — $BA/186 TXS — $9A/154

ROZKAZY MODYFIKUJĄCE ZAWARTOŚĆ REJESTRÓW INDEKSOWYCH

Rejestry indeksowe wykorzystywane są w programie maszynowym do konstruowania pętli, w których służą jako modyfikatory adresów przetwarzanych obszarów pamięci i jednocześnie jako licznik pętli. Podobną funkcję spełniają indeksy tablic w języku BASIC. Licznik każdej pętli programowej wymaga modyfikacji — zmniejszania lub zwiększania o wartość kroku po każdorazowym wykonaniu ciągu rozkazów zawartych w pętli. Mikroprocesor 6502 dostarcza programiście bardzo wygodnych rozkazów do realizacji tego zadania. Rozkazami tymi są:

INX — zwiększ o jeden zawartość rejestru X;
DEX — zmniejsz o jeden zawartość rejestru X;
INY — zwiększ o jeden zawartość rejestru Y;
DEY — zmniejsz o jeden zawartość rejestru Y;

Kody rozkazów:

INX — $E8/232 DEX — $CA/202 INY — $C8/200
DEY — $88/136

Każdy z tych rozkazów zajmuje jeden bajt pamięci i może być użyty wyłącznie z wewnętrznym trybem adresowania. Ich wykonanie może zmienić stan wskaźników N i Z rejestru flagowego. Do tej grupy rozkazów należałoby również zaliczyć dwa rozkazy przeznaczone do zmniejszania lub zwiększania o jeden zawartości bajtu z pamięci:

INC — zwiększ o jeden zawartość bajtu z pamięci o adresie wskazanym przez operand;
DEC — zmniejsz o jeden zawartość bajtu o adresie wskazanym przez operand;

Dzięki nim w łatwy sposób można wykorzystać bajt z pamięci jako licznik pętli programowej.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
INC			
absolutne	INC $D300	$EE/238	3
absolutne indeksowane X	INC $D300, X	$FE/254	3
strony zerowej	INC 240	$E6/230	2
strony zerowej indeks. X	INC 240, X	$F6/246	2
DEC			
absolutne	DEC $D300	$CE/206	3
absolutne indeksowane X	DEC $D300, X	$DE/222	3
strony zerowej	DEC 240	$C6/198	2
strony zerowej indeks. X	DEC 240, X	$D6/214	2

Rozkazy INC i DEC mogą zmienić stan wskaźników N i Z rejestru flagowego.

ROZKAZY PORÓWNUJĄCE ZAWARTOŚĆ REJESTRÓW A, X, Y Z ZAWARTOŚCIĄ PAMIĘCI

Rozkazy z tej grupy odejmują zawartość bajtu z pamięci lub (gdy użyte są z natychmiastowym trybem adresowania) wartość operandu od zawartości rejestru. W zależności od wyniku odejmowania zmieniają one stan wskaźników N, Z i C.
N — ustawiany jest, gdy bajt w pamięci jest większy od zawartości rejestru;
Z — ustawiany jest, gdy porównywane bajty są równe;
C — ustawiany jest, gdy bajt w pamięci jest mniejszy od zawartości rejestru;
Należy zaznaczyć, że rozkazy te nie zmieniają zawartości porównywanych bajtów. Rozkazami tymi są:

CMP — porównaj zawartość akumulatora z zawartością bajtu z pamięci;

CPX — porównaj zawartość rejestru X z zawartością bajtu z pamięci;

CPY — porównaj zawartość rejestru Y z zawartością bajtu z pamięci;

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
CMP			
natychmiastowe	CMP #14	$C9/201	2
absolutne	CMP 3000	$CD/205	3
absolutne indeksowane X	CMP 3000,X	$DD/221	3
absolutne indeksowane Y	CMP 3000,Y	$D9/217	3
strony zerowej	CMP $A9	$C5/197	2
strony zerowej indeks. X	CMP $A9,X	$D5/213	2
indeksowane pośrednie	CMP ($A9,X)	$C1/193	2
pośrednie indeksowane	CMP ($A9,Y)	$D1/209	2
CPX			
natychmiastowe	CPX #14	$E0/224	2
absolutne	CPX 3000	$EC/236	3
strony zerowej	CPX $A9	$E4/228	2
CPY			
natychmiastowe	CPY #14	$C0/192	2
absolutne	CPY 3000	$CC/204	3
strony zerowej	CPY $A9	$C4/196	2

ROZKAZY WYKONUJĄCE OPERACJE NA STOSIE MIKROPROCESORA

PHA — prześlij akumulator na stos i zmniejsz o jeden wskaźnik stosu;

PHP — prześlij rejestr flagowy na stos i zmniejsz o jeden wskaźnik stosu;

PLA — prześlij bajt ze szczytu stosu do akumulatora i zwiększ o jeden wskaźnik stosu;

PLP — prześlij bajt ze szczytu stosu do rejestru flagowego i zwiększ o jeden wskaźnik stosu;

Rozkazy te są rozkazami jednobajtowymi i mogą być użyte tylko z wewnętrznym trybem adresowania. PLA może zmienić stan wskaźników N i Z rejestru flagowego, natomiast PHP i PHA nie oddziałują na żadne wskaźniki. PLP może zmienić stan wszystkich wskaźników.
Kody rozkazów:

PHA	$48/72	PLA	$68/104
PHP	$08/8	PLP	$28/40

ROZKAZY WYKONUJĄCE OPERACJE ARYTMETYCZNE I LOGICZNE

ADC — dodaj zawartość bajtu z pamięci lub wartość operandu (adresowanie natychmiastowe) do zawartości akumulatora. Wynik dodawania pozostaje w akumulatorze. Wynik dodawania może być zwiększony o jeden, gdy przed wykonaniem rozkazu ustawiony był wskaźnik C. Dlatego zawsze przed użyciem tego rozkazu należy zerować wskaźnik C (rozkaz CLC), chyby że wymagana jest korekta wyniku (dodawanie liczb binarnych zapisanych na kilku bajtach). ADC może zmienić stan wskaźników N, V, Z i C rejestru flagowego.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
natychmiastowe	ADC #$80	$69/105	2
absolutne	ADC $D300	$6D/109	3
absolutne indeksowane X	ADC $D300,X	$7D/125	3
absolutne indeksowane Y	ADC $D300,Y	$79/121	3
strony zerowej	ADC 240	$65/101	2
strony zerowej indeks X	ADC 240,X	$75/117	2
indeksowane pośrednie	ADC (240,X)	$61/97	2
pośrednie indeksowane	ADC (240),Y	$71/113	2

AND — wykonaj iloczyn logiczny (AND) zawartości bajtu z pamięci i zawartości akumulatora. Rozkaz porównuje kolejno wszystkie bity bajtu z pamięci i akumulatora. Gdy porównywane bity są równe jeden, to odpowiadający im bit wyniku jest również równy jeden. W każdym innym przypadku bit wyniku jest zerem. Wynik rozkazu pozostaje w akumulatorze.
Przykład:

```
bit          7 6 5 4 3 2 1 0
             1 0 1 1 1 1 0 1   bajt z pomięci
AND          0 0 0 0 1 1 0 0   bajt z akumulatora
             ---------------
             0 0 0 0 1 1 0 0   wynik w akumulatorze
```

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
natychmiastowe	AND #$80	$29/41	2
absolutne	AND $D300	$2D/45	3
absolutne indeksowane X	AND $D300,X	$3D/61	3
absolutne indeksowane Y	AND $D300,Y	$39/57	3
strony zerowej	AND 240	$25/37	2
strony zerowej indeks. X	AND 240,X	$35/53	2
indeksowane pośrednie	AND (240,X)	$21/33	2
pośrednie indeksowane	AND (240),Y	$31/49	2

AND oddziałuje na wskaźniki N i Z rejestru flagowego.

ASL — przesuń zawartość bajtu z akumulatora lub z pamięci o jedną pozycję bitową w lewo. Najbardziej znaczący bit przesuwanego bajtu przenoszony jest do wskaźnika C, bit najmniej znaczący jest zerowany. Wykonywana operacja przesuwania w lewo odpowiada mnożeniu zawartości bajtu przez dwa. ASL oddziałuje na wskaźnik N, Z i C.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
absolutne	ASL $D300	$0E/14	3
absolutne indeksowane X	ASL $D300,X	$1E/30	3
strony zerowej	ASL 240	$06/6	2
strony zerowej indeks. X	ASL 240,X	$16/22	2
akumulatora	ASL A	$0A/10	1

BIT — wykonaj iloczyn logiczny zawartości bajtów z pamięci i akumulatora. Rozkaz ten działa podobnie jak rozkaz AND, z tą tylko różnicą, że nie zmienia zawartości porównywanych bajtów i oddziałuje na inne wskaźniki rejestru F. BIT zapala wskaźnik Z, gdy bajty są równe, ponadto do wskaźnika N kopiowany jest siódmy bit bajtu z pamięci, a do wskaźnika V — szósty bit tego bajtu. Na stan pozostałych wskaźników BIT nie oddziałuje.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
absolutne	BIT $D300	$2C/44	3
strony zerowej	BIT 240	$24/36	2

EOR — wykonaj logiczną różnicę symetryczną (exclusive OR) na zawartościach bajtu z pamięci i akumulatora. Rozkaz ten porównuje kolejno bity bajtu z pamięci i akumulatora. Jeże-

li porównywane bity są jedynkami, to odpowiadający im bit wyniku jest zerem; gdy są zerami to odpowiadający im bit wyniku jest również zerem; natomiast w sytuacji, gdy przynajmniej jeden z porównywanych bitów jest jedynką, to odpowiadający im bit wyniku jest jedynką.

Przykład:

```
bit          7 6 5 4 3 2 1 0
             1 0 1 1 1 1 0 1   bajt z pamięci
       EOR   0 0 0 0 1 1 0 0   bajt z akumulatora
             ---------------
             1 0 1 1 0 0 0 1   wynik w akumulatorze
```

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
natychmiastowe	EOR #$80	$49/73	2
absolutne	EOR $D300	4D/77	3
absolutne indeksowane X	EOR $D300,X	$5D/93	3
absolutne indeksowane Y	EOR $D300,Y	$59/89	3
strony zerowej	EOR 240	$45/69	2
strony zerowej indeks. X	EOR 240,X	$55/85	2
indeksowane pośrednie	EOR (240,X)	$41/65	2
pośrednie indeksowane	EOR (240),Y	$51/81	2

LSR — przesuń zawartość bajtu z pamięci lub z akumulatora o jedną pozycję bitową w prawo. Najmniej znaczący bit przesuwanego bajtu przenoszony jest do wskaźnika C, bit najbardziej znaczący jest zerowany. Wykonywana operacja przesuwania w prawo odpowiada dzieleniu zawartości bajtu przez dwa. LSR oddziałuje na wskaźnik N, Z i C. Ustawiony wskaźnik C świadczy o powstałej w wyniku dzielenia reszcie.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
absolutne	LSR $D300	$4E/78	3
absolutne indeksowane X	LSR $D300,X	$5E/94	3
strony zerowej	LSR 240	$46/70	2
strony zerowej indeks. X	LSR 240,X	$56/86	2
akumulatora	LSR A	$4A/74	1

ORA — wykonaj sumę logiczną (OR) na zawartościach bajtów z pamięci i akumulatora. Rozkaz porównuje kolejno wszystkie bity bajtów z pamięci i akumulatora. Jeżeli porównywane bity są zerami, to odpowiadający im bit wyniku jest również zerem. Dla każdej innej kombinacji porównywanych bitów bit wyniku jest jedynką. Wynik operacji pozostawiony jest w akumulatorze.

Przykład:

```
bit          7 6 5 4 3 2 1 0
             1 0 1 1 1 1 0 1   bajt z pamięci
       EOR   0 0 0 0 1 1 0 0   bajt z akumulatora
             ---------------
             1 0 1 1 1 1 0 1   wynik w akumulatorze
```

Rozkaz może zmienić stan wskaźników N i Z rejestru F.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
natychmiastowe	ORA #$80	$09/9	2
absolutne	ORA $D300	$0D/13	3
absolutne indeksowane X	ORA $D300,X	$1D/29	3
absolutne indeksowane Y	ORA $D300,Y	$19/25	3
strony zerowej	ORA 240	$05/5	2
strony zerowej indeks. X	ORA 240,X	$15/21	2
indeksowane pośrednie	ORA (240,X)	$01/1	2
pośrednie indeksowane	ORA (240),Y	$11/17	2

ROL — „obróć" zawartość bajtu z pamięci lub akumulatora o jedną pozycję w lewo. Rozkaz działa podobnie jak ASL. Zawartość „obracanego" bajtu przesuwana jest o jedną pozycję bitową w lewo, przy tym do bitu zerowego wprowadzana jest zawartość wskaźnika C (ASL zeruje ten bit), a bit siódmy kopiowany jest do wskaźnika C i stanowi jego nową wartość. Rozkaz ROL wspólnie z rozkazem ASL używane są do mnożenia przez dwa liczb zapisanych na kilku bajtach. Poza wskaźnikiem C rozkaz ROL może zmienić stan wskaźników N i Z.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
absolutne	ROL $D300	$2E/46	3
absolutne indeksowane X	ROL $D300,X	$3E/62	3
strony zerowej	ROL 240	$26/38	2
strony zerowej indeks. X	ROL 240,X	$36/54	2
akumulatora	ROL A	$2A/42	1

ROR — „obróć" zawartość bajtu z pamięci lub akumulatora o jedną pozycję bitową w prawo. Rozkaz działa podobnie jak LSR. Zawartość „obracanego" bajtu przesuwana jest o jedną pozycję bitową w prawo, przy tym do bitu siódmego wprowadzana jest zawartość wskaźnika C (LSR zeruje ten bit), a bit zerowy kopiowany jest do wskaźnika C i stanowi jego nową wartość. Rozkaz ROR wspólnie z rozkazem LSR używane są do dzielenia przez dwa liczb zapisanych na kilku bajtach. Poza wskaźnikiem C rozkaz ROR może zmienić stan wskaźników N i Z.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
absolutne	ROR $D300	$6E/110	3
absolutne indeksowane X	ROR $D300,X	$7E/126	3
strony zerowej	ROR 240	$66/102	2
strony zerowej indeks. X	ROR 240,X	76/118	2
akumulatora	ROR A	$6A/106	1

SBC — odejmij zawartość bajtu z pamięci lub wartość operandu (adresowanie natychmiastowe) do zawartości akumulatora. Wynik odejmowania pozostaje w akumulatorze. Wynik ten może być zmniejszony o jeden, gdy przed wykonaniem rozkazu zgaszony był (równy zero) wskaźnik C. Dlatego zawsze przed użyciem tego rozkazu należy „zapalić" wskaźnik C (rozkaz SEC). SBC może zmienić stan wskaźników N, V, Z i C rejestru flagowego.

dostępne tryby adresowania	format rozkazu	kod hex/dzies.	l. zaj. bajtów
natychmiastowe	SBC #$80	$E9/233	2
absolutne	SBC $D300	$ED/237	3
absolutne indeksowane X	SBC $D300,X	$FD/253	3
absolutne indeksowane Y	SBC $D300,Y	$F9/249	3
strony zerowej	SBC 240	$E5/229	2
strony zerowej indeks. X	SBC 240,X	$F5/245	2
indeksowane pośrednie	SBC (240,X)	$E1/225	2
pośrednie indeksowane	SBC (240),Y	$F1/241	2

POZOSTAŁE ROZKAZY

BRK — rozkaz ten generuje przerwanie programowe i jest stosowany przy testowaniu poprawności działania skonstruowanego programu maszynowego. W testowanym programie umieszcza się, w odpowiednich miejscach, rozkaz BRK. Gdy zostanie on wykonany realizacja testowanego programu zostanie przerwana. Wówczas za pomocą odpowiedniej procedury monitorującej można „obejrzeć" zawartość rejestrów mikroprocesora i określonych obszarów pamięci, oceniając na ich podstawie poprawność działania wykonanego programu bądź jego fragmentu. Rozkaz ten zapala wskaźnik B rejestru F. Rozkaz BRK zajmuje jeden bajt pamięci i może być stosowany jedynie z wewnętrznym trybem adresowania. Jego kod: $00/0.

NOP — nic nie rób. Rozkaz ten zatrzymuje na okres dwóch cykli zegarowych pracę mikroprocesora. Może być stosowany do budowania pętli opóźniających lub można nim zastąpić, po zakończeniu testowania, rozkazy BRK wstawione w tym celu w program maszynowy. NOP zajmuje jeden bajt pamięci, nie oddziałuje na żadne wskaźniki rejestru F i może być użyty tylko z wewnętrznym trybem adresowania. Jego kod: $EA/234.

RTI — „wróć z przerwania". Rozkaz ten służy jako „wyjście" z procedur obsługujących niektóre rodzaje przerwań mikroprocesora. Odtwarza on, zapamiętane na stosie, rejestr F i licznik programu i aktualizuje wskaźnik stosu. RTI zajmuje jeden bajt pamięci i musi być używany tylko z wewnętrznym trybem adresowania. Jego kod: $40/64.

RTS — „wróć z podprogramu". Rozkaz ten służy jako „wyjście" z podprogramów wywoływanych rozkazem JSR. Odtwarza on, zapamiętany na stosie, licznik programu i zwiększa wartość wskaźnika stosu o dwa. Nie oddziałuje na żadne wskaźniki rejestru F. Zajmuje jeden bajt pamięci i jest dostępny tylko z wewnętrznym trybem adresowania. Jego kod: $60/96.

Umieszczone poniżej przykłady podprogramów powinny ułatwić zrozumienie zasad tworzenia procedur w języku maszynowym. W tym celu kody źródłowe tych procedur opatrzone zostały licznymi komentarzami. Ostatni program, napisany w BASIC-u, przeznaczony jest do tworzenia linii DATA, zawierających powstały w wyniku asemblacji kod maszynowy zapisany na taśmie lub dysku. Utworzone linie DATA program ten również zapisuje na taśmę lub dysk. Można je dołączyć instrukcją ENTER do tworzonego programu w języku BASIC.

H. KRASUSKI

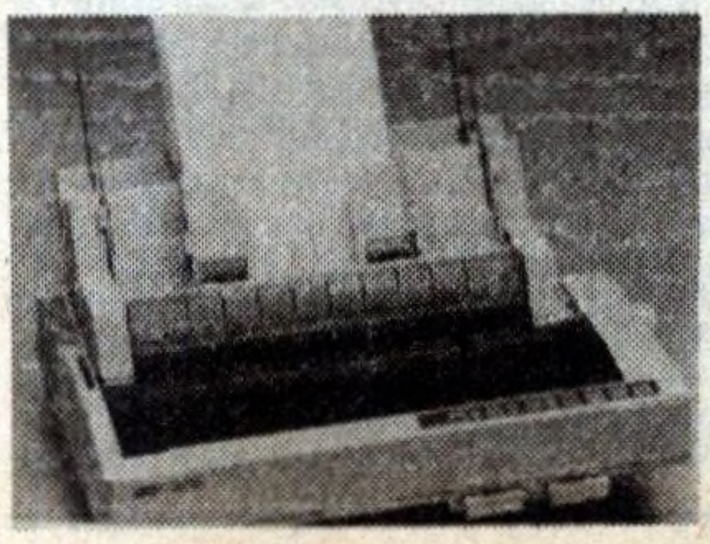
Bibliografia:
1. ZAKS Rodney: Programming the 6502, 6502 Series-Volume. SYBEX Inc., 1980 r.
2. LAVENTHAL L.A.: 6502 Assembly Language Programming. Osborne. McGraw-Hill Inc., 1979 r.


```
0100         .OPT OBJ
0110 ;Ta linia zawiera informacje dla assemblera
0120         *= $0600    ;od tego adresu
0130 ;asembler umiesci kod wynikowy podprogramu
0140 ;*************************************************
0150 ;          Podprogram 1                          *
0160 ;*************************************************
0170 ;podprogram umozliwia   odczyt/zapis z   dyskietki
0180 ;sektora o dowolnym numerze.
0190 ;Sposob wywolania z jezyka BASIC:
0200 ;   X=USR(ADRES,NRD,FUN,ADR(SEK$),NRSEK)
0210 ;gdzie:
0220 ;NRD     - zmienna zawierajaca nr drive'u
0230 ;FUN     - kod funkcji do wykonania
0240 ; FUN=82  odczyt sektora
0250 ; FUN=87  zapis sektora
0260 ;SEK$    - zmienna tekstowa o dlugosci 128 bajtow
0270 ;z/do ktorej zapisywany/odczytywany jest sektor
0280 ;NRSEK   - zmienna zawierajaca nr sektora
0290 ;do odczytu/zapisu
0300 ;Po wykonaniu podprogramu zmienna X zawiera 1
0310 ;gdy odczyt/zapis zakonczyl sie pomyslnie
0320 ;kod bledu w przeciwnym wypadku
0330 ;
0340 ;deklaracja zmiennych dla programu maszynowego
0350 ;
0360 NRD     = $0301 ;nr drive'u
0370 FUN     = $0302 ;kod funkcji
0380 STATUS  = $0303 ;wynik wykonania
0390 SEKM    = $0304 ;adres SEK$ -mlodszy bajt
0400 SEKS    = $0305 ;adres SEK$ -starszy bajt
0410 NRSM    = $030A ;nr sektora -mlodszy bajt
0420 NRSS    = $030B ;nr sektora -starszy bajt
0430         PLA     ;usun ze stosu liczbe param.
0440         PLA     ;pobierz ze stosu st. bajt NRD
0450         PLA     ;pobierz ze stosu ml. bajt NRD
0460         STA NRD ;zapamietaj go w zmiennej NRD
0470         PLA     ;zrob to samo z
0480         PLA     ;kodem funkcji do wykonania
0490         STA FUN ;zapamietaj kod funkcji w FUN
0500         PLA     ;st.bajt adresu SEK$
0510         STA SEKS ;zapamietaj w SEKS
0520         PLA      ;ml. bajt tego adresu
0530         STA SEKM ;zapamietaj w SEKM
0540         PLA      ;st. bajt numeru sektora
0550         STA NRSS ;zapamietaj w NRSS
0560         PLA      ;a mlodszy bajt
0570         STA NRSM ;zapamietaj w NRSM
0580         JSR $E453  ;skocz do procedury SIOV
0590         LDA STATUS ;wynik zap/odcz.do akumulatora
0600         STA $D4    ;zapamietaj wynik w bajtach
0610         LDA #0     ;$D4 i $D5 strony zerowej
0620         STA $D5
0630         RTS        ;wyjdz z podprogramu
0640         .END

10 ;*************************************************
20 ;          Podprogram 2                          *
30 ;*************************************************
40 ;Podprogram zamienia komunikaty /stale i zmienne
50 ;tekstowe/ z trybu "normalnego" na "invers video"
60 ;i odwrotnie. UWAGA: dlugosc komunikatu
70 ;nie moze przekraczac 255 bajtow
80 ;Sposob wywolania z jezyka BASIC
90 ;   X=USR(ADRES,ADR(KOM$),LEN(KOM$))
0100 ;gdzie:
0110 ;KOM$ stala lub zmienna tekstowa, ktora ma byc
0120 ;zmieniona
0130 ;
0140     PLA          ;usun ze stosu liczb. param.
0150     PLA          ;pob. ze stosu st. bajt adresu
0160     STA 204      ;KOM$ i zapam. go w bajcie 204
0170     PLA          ;a mlodszy bajt tego adresu
0180     STA 203      ;zapamietaj w bajcie 203
0190     PLA          ;dlugosc stalej /tylko jeden
0200     PLA          ;bajt/ zapamietaj w bajcie
0210     STA 205      ;205
0220     LDY #0       ;zero do rejestru Y
0230 ZAM LDA (203),Y  ;zaladuj kolejny bajt stalej
0240 ;                KOM$ do akumulatora
0250     EOR #$80     ;zmien tryb
0260     STA (203),Y  ;przeslij zmieniony bajt
0270 ;                z powrotem na jego miejsce
0280     INY          ;zwieksz o jeden liczbe zmie-
0290 ;                nionych bajtow
0300     CPY 205      ;czy zamienione wszystkie
0310     BNE ZAM      ;nie, to zamien kolejny
0320     RTS          ;tak, to wyjdz z podprogramu
0330     .END

10 ;*************************************************
20 ;          Podprogram 3                          *
30 ;*************************************************
40 ;Podprogram przesyla dane tworzace obraz
50 ;z dowolnego obszaru pamieci do obszaru
60 ;pamieci ekranu
62 ;tryb w jakim wyswietlany jest obraz musi byc
64 ;ustawiony z poziomu jezyka BASIC instr. GRAPHIC
65 ;przed wywolaniem podprogramu
90 ;Sposob wywolania z jezyka BASIC:
0100 ;     X=USR(ADRES,ADR(DANE0$),LB)
0110 ;gdzie:
0120 ;DANE0$ - stala tekstowa zawierajaca dane
0130 ;tworzace obraz
0135 ;LB - dlugosc DANE0$ w bajtach
0140 ;
0150 SKADM = $D6    ;mlodszy bajt adresu stalej DANE0$
0160 SKADS = $D7    ;starszy bajt tego adresu
0162 LBLOK = $D9    ;liczba blokow 256 bajtowych
0164 RESZT = $D8    ;reszta bajtow do przeslania
0166 ;(LBLOK*256)+RESZT = liczbie bajtow do przesl.
0170       PLA      ;usun ze stosu bajt z liczba param.
0180       PLA      ;pob. ze stosu st. bajt adresu
0190       STA SKADS;i zapamietaj go w SKADS
0200       PLA      ;pob. ze stosu ml. bajt adresu
0210       STA SKADM;zapamietaj go w SKADM
0212       PLA      ;pob. liczbe blokow
0213       STA LBLOK;zapamietaj ja w LBLOK
0214       PLA      ;pobierz reszte
0215       STA RESZT;zapamietaj ja w RESZT
0220       LDX LBLOK;liczba blokow do rej. X
0230       LDY #0   ;zero do rej. Y
0240 P1    LDA (SKADM),Y ;pob. kolejny bajt z DANE0$
0250       STA (88),Y    ;zapamietaj go w obszarze
0260 ;                pamieci ekranu
0270       DEY      ;czy przeslany caly blok
0280       BNE P1   ;nie, to skocz do etykiety P1
0290       INC SKADS;tak, zwiekasz o 1 st. bajt adresu
0300 ;                DANE0$
0310       INC 89   ;zwiekasz o jeden starszy bajt
0320 ;                adresu pamieci ekranu
0330       DEX      ;odejmij 1 od liczby blokow
0335       BMI KON  ;zakoncz przesylanie
0340       BNE P1   ;skocz do P1, gdy liczba blokow
0350 ;                nie jest zerem
0352       LDY RESZT;reszta do rej. Y
0354       BNE P1   ;gdy RESZT<>0 to przesylaj
0360 KON   RTS      ;gdy zero wyjdz z podprogramu
0370       .END
0380 ;UWAGA: podprogram ten bardzo szybko zmienia
0390 ;i wyswietla ekrany w trybie graficznym.
0400 ;Jezeli uzywa sie go do wyswietlania ekranow
0410 ;w trybie tekstowym, to DANE0$ musi zawierac
0420 ;kody PEEK/POKE wyswietlanych znakow a nie
0430 ;ich kody ATASCII.
```

```
1 REM ********************************
2 REM *      PROGRAM 4               *
3 REM ********************************
10 ? CHR$(125)
20 ? "Program tworzy linie DATA z kodu"
30 ? "maszynowego zapisanego na tasmie"
40 ? "lub dysku. Zbior z kodem
maszynowym"
50 ? "musi miec format wymagany przez
DOS"
100 DIM CIO$(6),BUF$(10000),LINIA$(92),
F$(14),KOD$(3)
110 CIO$(1)=CHR$(104):CIO$(2)=CHR$(162)
:CIO$(3)=CHR$(32)
120 CIO$(4)="L":CIO$(5)="V":CIO$(6)=CHR
$(228)
130 TRAP 130:? "Podaj urzadzenie i nazwe
"
131 ? "zbioru z kodem maszynowym: "
:INPUT F$
140 IF F$(1,2)<>"C:" AND F$(1,2)<>"D:"
THEN ? CHR$(253):GOTO 130
150 TRAP 2000:CLOSE #2:OPEN #2,4,0,F$
160 IOCB=864:POKE IOCB+2,7
165 SB=INT(ADR(BUF$)/256):MB=ADR(BUF$)-
SB*256
170 POKE IOCB+4,MB:POKE IOCB+5,SB
175 POKE IOCB+8,0:POKE
IOCB+9,PEEK(742)-PEEK(145)
180 X=USR(ADR(CIO$)):CLOSE #2:TRAP
40000
185 AD=ADR(BUF$)
190 IF PEEK(AD)<>255 AND PEEK(AD+1)
<>255 THEN 2100
195 AP=PEEK(AD+3)*256+PEEK(AD+2)
200 AK=PEEK(AD+5)*256+PEEK(AD+4)
205 IF AK-AP+1>10000 THEN GOTO 2200
210 ? :? CHR$(253);CHR$(253);"zbior
";F$;" OK"
215 TRAP 215:? :? "Podaj urzadzenie i
nazwe zbioru"
220 ? "do ktorego beda wpisane linie
DATA"
225 F$="":INPUT F$
230 IF F$(1,2)<>"C:" AND F$(1,2)<>"D:"
THEN ? CHR$(253):GOTO 215
236 ? "podaj poczatkowy nr linii DATA:
":INPUT NRL
237 ? "podaj krok numeracji: ":INPUT
KROK
239 TRAP 2300:CLOSE #2:OPEN
#2,8,0,F$:TRAP 40000
240 FOR I=0 TO AK-AP STEP 20
245   FOR J=1 TO 92:LINIA$(J)=" ":NEXT
J
250   J=LEN(STR$(NRL))
255   LINIA$(1,J)=STR$(NRL):J=J+2:
LINIA$(J)="DATA":J=J+5
260   FOR K=0 TO 19
270     N=PEEK(AD+6+I+K):KOD$="000":KOD$
(4-LEN(STR$(N)))=STR$(N)
275     LINIA$(J)=KOD$:J=J+3
280     IF K<>19 THEN
LINIA$(J)=",":J=J+1
295   NEXT K
300   LINIA$(J)=CHR$(155):PRINT
#2;LINIA$:NRL=NRL+KROK:NEXT I
305 CLOSE #2
310 ? CHR$(253);CHR$(253)
315 ? :? "Gotowe..."
320 ? "Chcesz utworzyc nastepny zbior
(T/N)"
325 CLOSE #1:OPEN #1,4,0,"K:":GET #1,K
330 IF K<>84 THEN END
335 CLR :GOTO 20
2000 ? CHR$(253);"BLAD NR: ";PEEK(195);"
PRZY ODCZYCIE ZBIORU ";F$
2010 CLR :GOTO 20
2100 ? CHR$(253);"ZBIOR ";F$;" NIE JEST
W FORMACIE DOS-a"
2102 ? "NAGLOWEK: ";PEEK(AD);" ";PEEK
(AD+1);" ";
2104 ? PEEK(AD+2);" ";PEEK(AD+3);" ";
PEEK(AD+4);" ";PEEK(AD+5)
2110 CLR :GOTO 20
2200 ? CHR$(253);"ZBIOR ";F$;" ZAJMUJE
WIECEJ NIZ 10000 BAJTOW"
2210 CLR :GOTO 20
2300 ? CHR$(253);"BLAD NR ";PEEK(195);"
PRZY ZAPISIE ZBIORU ";F$
2310 CLR :GOTO 20
```



KTO POMOŻE?

Czy stara, wysłużona Odra 1305 może uchodzić za symbol nowoczesności? — Pytanie retoryczne i dlatego kierownictwo Wojskowych Zakładów Uzbrojenia nr 1 w Krakowie zaczęło rozglądać się za czymś nowszym. Oczywiście nie po to tylko, by mieć nowy symbol. — *Widzimy szansę przejścia „do przerobu„ przez nasz ośrodek wielu prac planistycznych i obliczeniowych. — Co na tym zyskają zakłady? — Przede wszystkim zmniejszymy zatrudnienie w administracji* — zapewnia dyrektor płk Ireneusz **Rosiak**. A nie jest to błacha sprawa, ponieważ każdy pracownik kosztuje zakłady 650 tys. zł rocznie. Na tę sumę składa się pensja, różne składki, obciążenia, podatki — i to bez względu na to, czy ten ktoś pracuje, czy też — przykładowo — przebywa na zwolnieniu lekarskim. — *A komputer nie choruje i jest nadzwyczaj dyspozycyjny* — tę prawdę krakowianie odkryli już dawno.

Oprócz redukcji administracji możliwe jest zwiększenie efektywności gospodarki materiałowej i kadrowej. Lepsze wykorzystanie dwóch najważniejszych czynników produkcji w zakładzie, to dodatkowy zysk... Tak się bowiem składa, że z maszyn i urządzeń niewiele da się już „wycisnąć", ponieważ wykorzystanie czasu pracy maszyn wynosi 96 proc.!

Ale to nie wszystko. Dyrekcja widzi możliwość „zaprzęgnięcia" komputera do planowania produkcji. Nie chodzi przy tym o sporządzanie prostych zestawień, lecz o coś na pograniczu fantastyki — oczywiście w naszych krajowych wyrunkach. Tym „czymś" byłyby dodatkowe korzyści wynikające nie z produkcji w ogóle, lecz z produkcji wykonanej w określonym czasie. Co to daje? — Lepiej jest wykorzystywany potencjał przedsiębiorstwa i zadowoleni klienci, ponieważ np. remont sprzętu odbywa się w dogodnych terminach dla odbiorcy.

Wszystko to być może... Póki co, nie ma jednak chętnych do „skomputeryzowania" zakładów. Podstawową przeszkodą jest to, że przedsiębiorstwo wykonuje produkcję dla potrzeb sił zbrojnych, a więc występuje problem tajemnicy wojskowej i państwowej. Ale to jeszcze nie wszystko. Krakowianie nie byliby sobą, gdyby nie liczyli pieniędzy. Dyrekcja ma więc następujący pogląd na sprawę: komputeryzacja — tak, ale... I tu następuje lista życzeń: muszą być oszczędności na funduszu płac, powinien ulec przyspieszeniu obieg informacji, kolejna sprawa, to eliminacja niektórych prac i dokumentów, udoskonalenie ewidencji materiałowej i zmniejszenie poziomu zapasów.

Warto zwrócić również uwagę i na to, że w WZU dostrzegane są dwie grupy korzyści. Jedne — wymierne — dadzą się wyliczyć jeszcze przed uruchomieniem systemu. I pewnie właśnie one zadecydują o jego wyborze i wprowadzeniu. Ale pomysłodawcy liczą również na korzyści, które w żaden sposób nie dadzą się obliczyć, ani oszacować ex ante. Niewymierne teraz, powinny jednak w przyszłości powiększyć masę zysku. Wynikać będą przede wszystkim z ogólnej poprawy organizacji pracy w przedsiębiorstwie i trafności decyzji podejmowanych w oparciu o dokładniejsze dane. Raz może to być potrzeba nowych uruchomień, innym zaś razem podjęcie produkcji deficytowych części zamiennych lub zabiegi o wykonanie usług dla kontrahenta zagranicznego. Tych możliwości może oczywiście być o wiele więcej.

Mógłby ktoś powiedzieć, że problem przed którym stanęły Wojskowe Zakłady Uzbrojenia nr 1 w Krakowie jest ich własnym kłopotem. Przecież jest to samodzielne, samorządne i samofinansujące się przedsiębiorstwo, więc niech sobie jakoś samo radzi. Ale ten nieobcy w naszej rzeczywistości społecznej sposób myślenia nie uwzględnia właśnie społecznych korzyści płynących z rozumnego traktowania problemów gospodarczych. Bo rzecz nie dotyczy tylko tej jednej sprawy w jednym z wielu przedsiębiorstw i stąd pytanie w tytule, adresowane do czytelników w mundurach.

Andrzej MĘDYKOWSKI

LIGA MYŚLĄCYCH

ZADANIE 1

Kosiarze mieli skosić dwie łąki. Rano wszyscy kosiarze zaczęli kosić większą łąkę, a po obiedzie podzielili się. Połowa pozostała i kosiła pierwszą łąkę, którą do wieczora skosili, druga zaś połowa poszła kosić drugą łąkę, której powierzchnia równała się połowie powierzchni pierwszej łąki. Ilu było kosiarzy, skoro wiemy, że resztę drugiej łąki skosił jeden kosiarz w ciągu następnego dnia?

ZADANIE 3

Kolumna pojazdów o długości $l = 2$ km porusza się z prędkością $V_k = 20$ km/h. Od czoła kolumny do ostatniego pojazdu i z powrotem jedzie łącznik na motocyklu z prędkością $V_t = 60$ km/h. Ile czasu potrzebuje łącznik na przebycie drogi tam i z powrotem oraz jaką drogę przebędzie w tym czasie?

ZADANIE 2

W pierwszym półroczu roku 1987 produkcja spadła o 12,2 proc. w stosunku do produkcji pierwszego półrocza roku 1986, liczba pracowników zaś zwiększyła się o 8,3 proc. O ile procent spadła wydajność jednego pracownika?

ZADANIE 4

Z trzech naczyń, w kształcie sześcianów, pierwsze jest o 1 dm wyższe niż drugie, a drugie o 1 dm wyższe od trzeciego. Kiedy drugie naczynie napełnimy wodą z pierwszego naczynia, a trzecie wodą z drugiego, to w pierwszym naczyniu będzie o 12 litrów wody więcej niż w drugim naczyniu. Jakie są wymiary naczyń?

ZADANIE 5

Podczas pierwszej jazdy samochodem zużyto 20 proc. benzyny, która znajdowała się w zbiorniku paliwa. Podczas drugiej jazdy zużyto 10 proc. ilości benzyny, która pozostała w zbiorniku po pierwszej jeździe. Po dwóch jazdach pozostało w zbiorniku 9 litrów benzyny. Ile litrów benzyny znajdowało się w zbiorniku przed pierwszą jazdą?

Rozwiązania zadań prosimy przesyłać do redakcji do końca czerwca br. z dopiskiem „Liga Myślących". Punktacja zależy od liczby prawidłowych rozwiązań. Wśród uczestników rozlosujemy książki, a na zwycięzców „Ligi" czekają dodatkowe nagrody.

OCALONY

Robert JASTRZĘBSKI

Nastąpił drugi dzień po wybuchu. Szedłem przez ogromną równinę w poszukiwaniu przedmiotów, którym wybuch nadał fantastyczne, a zarazem przerażające kształty. Ciągle potykałem się o jakieś stopniowe elementy metalowe, pochodzące prawdopodobnie z konstrukcji budowlanych miasta, jakie istniało tu przed eksplozją. Przez gęstą warstwę pyłów powybuchowych i zgęstniałych chmur z trudem przebijało się szare i blade światło słoneczne.

Uparcie szedłem naprzód. Chciałem za wszelką cenę znaleźć jakieś schronienie i odszukać choćby jednego żyjącego człowieka. Głód dawał mi się we znaki. Po godzinie zaczęło się ściemniać. Pod szarym sklepieniem chmur przesuwały się wolno czarne, kłębiaste warstwy dziwnego pyłu. Skierowałem się w stronę odległych wzgórz.

W pewnej chwili na jednym z wzniesień zobaczyłem oślepiający błysk. Zjawisko to powtórzyło się jeszcze trzykrotnie. Pomyślałem, że musi być tam człowiek. Zacząłem biec. Po parominutowym biegu upadłem. Byłem bardzo zmęczony. W oddali znowu zajaśniało ostre światło. Rozbłysnęło trzy razy i zgasło. Poderwałem się z ziemi.

Po pewnym czasie z trudem dotarłem do pierwszego wzgórza. Było nagie, usłane mnóstwem skurczonych kamieni. W niektórych miejscach jaśniały szkliste stopione skały. Zacząłem się wspinać. Raz po raz ziemia wymykała mi się spod nóg. Ręce pokaleczyłem sobie o okruchy skalne. Chciałem za wszelką cenę dotrzeć do źródła owego tajemniczego światła.

Wspinając się coraz wyżej, odczułem, że zrobiło się nieco chłodniej. Zdumiałem się, skąd te wahania temperatury. Odpowiedź przyszła niebawem. Z szarego sklepienia chmur runął potok deszczu. Ale nie był to zwykły deszcz. Krople były gęste, jakby zmieszane z pyłem. Sprawiało to wrażenie, że zamiast kropli wody, spadają z chmur krople oleju. Po upływie paru sekund mój podarty kombinezon pokrył się gęstą i lepką substancją.

Próbowałem dalej się wspinać, lecz ślizgałem się jak na lodzie. Traciłem równowagę i upadłem w płynącą po zboczu olejową ciecz. Podczas jednej z takich prób zacząłem zjeżdżać w dół zbocza. Zbocze urwało się i spadłem. Straciłem przytomność.

Gdy się obudziłem — było już prawie ciemno. Deszcz nie padał. Z ledwością odlepiłem się od podłoża. Oleista ciecz zgęstniała, cały byłem oblepiony szaro-brunatną masą. Głowa bolała mnie okropnie. Obolały i potłuczony powoli stanąłem na nogi. Rozejrzałem się. Półka skalna była mała i wąska. Poniżej było piętnastometrowe urwisko. O zejściu w dół nie było mowy. Wejście na górę też było niemożliwe.

Usiadłem zrozpaczony. Zapadła noc. Było bardzo zimno. Póki starczyło mi sił starałem się nie zasnąć. Lecz nie trwało to zbyt długo. Po parunastu minutach zmożył mnie sen.

Nagle odczułem wyraźną zmianę temperatury. Zrobiło się ciepło. Otworzyłem oczy. Obok mnie ze szczeliny skalnej wydobywał się jaskrawy promień świetlny. Po chwili znikł. Wstałem i podszedłem do szczeliny. Krzyknąłem do wnętrza. Odpowiedziało tylko echo. Chciałem krzyknąć jeszcze raz...

Oślepił mnie blask potężnej wiązki światła. Wszystko co dotychczas mnie otaczało, zaczęło znikać. Całą przestrzeń przeniknęło światło. Nie mogłem dojrzeć nawet własnej postaci. W czasie trwania tej jasności do moich uszu dobiegał cichy, a zarazem dziwny dźwięk. W zasadzie nie słyszałem tego uszami. Miałem wrażenie, że przenikał on bezpośrednio do mojego mózgu. Poczułem, że moje niewidzialne ciało przesuwa się. Nie potrafiłem określić, w jakim kierunku się przemieszczam, ale najwyraźniej odczuwałem ruch. Czułem, że przemieszczam się coraz szybciej. Jasność zaczęła się zmieniać w przestrzeń falującą całą gamą kolorów. Barwy wirowały jak szalone. Za jakiś czas zaczęły się oddalać. Całą przestrzeń powoli wypełniały pasma o ciemnych kolorach. Koncentrując się stworzyły nieprzenikalną ciemność.

Dźwięczenie ustało. Zewsząd otaczała mnie czerń. Czułem się bardzo rozluźniony, lekki, nie odczuwałem ciężaru swojego ciała. Nie wiem jak długo trwałem w tym stanie. Pojęcie czasu jakby przestało istnieć.

Później zacząłem odczuwać jak z tego dziwnego stanu przechodzę w stan snu. Powróciło poczucie własnego ciała.

Obudziłem się. Leżałem na dziwnej elastycznej srebrnej płycie zawieszonej w powietrzu. Płyta znajdowała się pośrodku wielkiej, białej sali. Nad moją głową wisiała wirująca, czerwona kula. Spojrzałem na siebie. Byłem ubrany w czarny jednolity kombinezon wykonany z połyskliwego materiału. Kombinezon nie miał żadnych kieszeni, ani zapięć. Stanowił z moim ciałem jakby całość. Nawet twarz miałem zasłoniętą, lecz wszystko widziałem i słyszałem bardzo dokładnie.

Wstałem z płyty i zacząłem się rozglądać. Wokół mnie nic nie było prócz białych ścian, lekko nachylonych ku górze. Żadnych okien, ani drzwi. Zacząłem iść przed siebie. Zauważyłem, że czerwona kula przesuwa się wraz ze mną.

Podszedłem do jednej ze ścian, wyciągnąłem rękę. Dłoń zniknęła w mlecznej przestrzeni. Cofnąłem rękę. Dłoń pojawiła się z powrotem. Wtem, coś mi kazało bym szedł naprzód. Wokół mnie nie było nikogo. Pomyślałem, by sprawdzić czy pozostałe ściany zachowują się w podobny sposób jak ta, przy której stoję. Chciałem się obrócić, gdy do mojego mózgu dotarł ponownie ten sam rozkaz.

Zacząłem się zastanawiać, kto ma tak potężną siłę, by wniknąć do mojego mózgu i przekazywać mi polecenia w postaci myśli. Spojrzałem do góry. Nademną wciąż wirowała czerwona kula. Domyślałem się, że to właśnie ona jest moim telepatycznym stróżem.

Ruszyłem naprzód. Przeniknąłem przez mglistą ścianę. Ukazała się tu podobna do poprzedniej sala, tyle że miała kolor niebieski. I znowu otrzymałem polecenie, żeby iść do przodu. Przechodziłem przez kolejne ściany i kolejne kolorowe sale.

Zaczęła mnie denerwować ta wędrówka. Stanąłem i spojrzałem na kulę. Nadal wirowała nad moją głową, pulsując czerwonym światłem. Podskoczyłem, by jej dotknąć. Była nieuchwytna. Zacząłem pytać, gdzie jestem, kim ona jest. Odpowiedzi nie było. Znowu kazano mi iść przed siebie.

Przekroczyłem kolejną ścianę, znalazłem się w ogromnym pomieszczeniu. Nie posiadało ono ani ścian, ani podłogi i sufitu. Wokoło otaczała mnie czarna przestrzeń. Spojrzałem na kulę. Zauważyłem jak zmienia swój kolor z czerwonego w żółty, potem w biały. Później stała się przezroczysta. Wtem, z miejsca, gdzie wirowała, strzelił w otchłań jaskrawy promień. Jednak zanim zniknęła, przekazała mi swój ostatni rozkaz, abym przygotował się na spotkanie z komputerem czasu.

W miejscu, gdzie zniknął promień, przejawił się jaskrawo świecący punkt. Punkt zaczął się powiększać. Blask był tak potężny, że zmusił mnie do zamknięcia oczu. Po paru sekundach odczułem, że świecenie zmalało. Otworzyłem oczy. Ujrzałem potężną błękitną kulę. Pulsowała ona tajemniczym blaskiem, jednakże nie mogłem nigdzie dojrzeć opuszczającego ją promieniowania.

Chciałem podejść do kuli, lecz nie mogłem zrobić kroku, chciałem coś powiedzieć, lecz nie mogłem wydusić z siebie słowa. Patrzyłem na kulę i nie mogłem oderwać od niej wzroku. Wtem ujrzałem na jej powierzchni skupiające się cienkie pasemka promieni świetlnych.

Nagle pasemka zniknęły w jednym punkcie. W moją stronę zaczęła zbliżać się wstęga światła. Wniknęła ona we mnie. Nic nie poczułem. Po pewnej chwili doznałem wrażenia, że błękitna kula mnie pochłania. Obraz, jaki dotychczas widziałem przed so-

bą zaczął się pokrywać gęstą mgłą. Po paru sekundach straciłem z oczu kulę...

Przede mną roztaczała się mleczna przestrzeń. Szum zaczął wzrastać. Niebawem był tak głośny, że trudno było wytrzymać. Myślałem, że stracę słuch. Nagle szum się urwał. Miałem uczucie, że zaczynam się przemieszczać. W miejsce mlecznej przestrzeni napłynęły niezrozumiałe obrazy. Nie mogłem dociec co one przedstawiały. Ich treść była jakaś fantastyczna, występowały na nich różnokolorowe elementy. Obrazy te powoli wirowały i nakładały się na siebie.

Z każdą chwilą prędkość wirowania rosła. Gdy prędkość była już nieskończenie wielka, cała ta dotychczas kolorowa mozaika zlała się w jedną ciemną otchłań...

Po chwili poczułem chłód. Doznałem wrażenia, że na czymś leżę. Zrobiło się ciepło. Ocknąłem się. Otworzyłem oczy, podniosłem głowę i rozejrzałem się dokoła. Leżałem znowu na półce skalnej, na którą spadłem, lecz zamiast otaczających mnie wtedy nagich skał, zewsząd otaczała mnie bujna roślinność. Spojrzałem w dół, w stronę równiny. Był to przepiękny widok. Tam, gdzie kiedyś były stopione i skruszone kamienie, teraz rosła zielona i soczysta trawa.

Wyłaniały się z niej potężne drzewa, pokryte listowiem olbrzymich rozmiarów. Gdzieniegdzie strzelały z liściastej gęstwiny wysokie, o fantastyczej formie rośliny, przypominające drzewa iglaste. Cała równina zdawała się być dziewiczym, zielonym oceanem.

Okolica wydawała mi się znajoma, ale jakże inaczej wyglądała ona przedtem, po eksplozji. Gdzie podziały się tajemnicze kule i pomieszczenie, w którym niedawno przebywałem? Ile czasu mogłem tam być? Czy to był fakt, czy tylko sen? Dręczyło mnie pytanie o pochodzenie tego tajemniczego komputera. Kto go stworzył? Był to komputer czasu — może więc to maszyna albo to „coś" przetransportowało mnie w czasie? Ale w przeszłość, czy w przyszłość?

Chyba nigdy nie znajdę na te pytania odpowiedzi. Spojrzałem na siebie. Ponownie byłem ubrany w swój stary i podarty kombinezon. Po tajemniczym, czarnym ubraniu nie było śladu. Przypomniałem sobie, że tutaj, w skale musi być szczelina, z której wydobywał się promień świetlny. Podszedłem do skały. Rozgarnąłem wiszące liście. Jednakże nic nie znalazłem, prócz kamiennej ściany, porośniętej mchem. Po szczelinie nie było śladu.

Odwróciłem się i podszedłem nad urwisko. Z łatwością zszedłem w dół po gąszczu korzeni i lian. Zanurzyłem się w wysokiej trawie i gęstym, wilgotnym listowiu. Podniosłem z ziemi solidnie wyglądający kij i zacząłem przedzierać się przez zieloną ścianę traw i krzewów. Słońce świeciło jaskrawym blaskiem, obdarzając swoim ciepłem i światłem zieloną równinę. Chwyciłem kij mocno w dłoń i ruszyłem przed siebie na spotkanie nowego świata.

PROGRAM SPECTRUM 49, 50

Dźwięki

```
  Program demonstruje dźwięki w
zakresie częstotliwości pomiędzy
8.18 Hz i 14080 Hz.
  Wysokość dźwięku określa się
bezpośrednio w Hz lub w skali od
-60 do +69.
  Dla dźwięków różniących się o
mniej niż 10% od częstotliwości
nominalnej program podaje nazwę
dźwięku.
  Program nie przyjmuje błędnych
danych, np. liter.

 100 PAPER 4: BORDER 4: CLS
 102 DIM p$(4): DIM t$(12,7): FO
R t=1 TO 12: READ t$(t,3 TO 5):
NEXT t: RESTORE
 104 DATA " c ","cis"," d ","dis
","  e ","  f ","fis"," g ","gis",
" a ","ais"," h "
 110 PRINT #0;AT 1,10; PAPER 7;
BRIGHT 1;"  Ton / Hz  "
 120 PAUSE 0: LET i$=INKEY$: GO
TO 130*(i$="h")+160*(i$="t")+120
*(i$<>"h" AND i$<>"t")
 130 INPUT TAB 8; PAPER 7; BRIGH
T 1;" 8.18< Hz <14080 "; LINE h$
: FOR n=1 TO LEN h$: IF NOT (h$(
n)>="0" AND h$(n)<="9" OR h$(n)=
".") THEN GO TO 130
 135 NEXT n: LET hz=VAL h$: IF h
z<8.18 OR hz>14080 THEN GO TO 13
0
 140 LET to=12*LN (hz/261.625565
)/LN 2
 150 GO TO 180
 160 INPUT TAB 8; PAPER 7; BRIGH
T 1;" -60< Ton < 69 "; LINE o$:
FOR n=1 TO LEN o$: IF NOT (o$(n)
>="0" AND o$(n)<="9" OR o$(n)=".
" OR o$(n)="-") THEN GO TO 160
 161 NEXT n
 165 LET to=VAL o$: IF to<-60 OR
 to>69 THEN GO TO 160
 170 LET hz=261.625565*2↑(to/12)
 179 REM wydruk i BEEP
 180 CLS : LET d=100+to: IF (d-I
NT to)<100.1 OR (d-INT to)>100.9
 THEN GO SUB 300
 190 LET s$=STR$ to:
     LET a$=p$+"Ton "+s$+p$
 200 PRINT AT 5,(32-LEN a$)/2;
PAPER 6; BRIGHT 1;a$
 210 LET s$=STR$ hz:
     LET b$=p$+s$+" Hz"+p$
 220 PRINT AT 17,(32-LEN b$)/2;
PAPER 6; BRIGHT 1;b$
 230 BORDER 6: BEEP .5,to: BORDE
R 4
 240 PAUSE 20:
 IF INKEY$<>"" THEN GO TO 110
 250 GO TO 230
 299 REM podprogram
 300 LET t=61+INT (to+.1)
 310 IF t>12 THEN LET t=t-12: GO
 TO 310
 320 PRINT AT 11,13; BRIGHT 1; P
APER 7; INK 2;t$(t): RETURN
9998 REM zapis na tasme
9999 CLEAR : SAVE "Ton/Hz" LINE
100: VERIFY "": GO TO 100
```

Klawiatura


```
 0>REM Program KLAWIATURA
 80 BORDER 4: CLS : BRIGHT 0
 90 FOR n=0 TO 7: READ bt: POKE
USR "a"+n,bt: NEXT n: REM a=©
 100 DATA 255,195,195,0,129,195,
231,255
 110 FOR n=60000 TO 60100: POKE
n,250: NEXT n
 120 FOR n=0 TO 16: READ m: POKE
 60000+m,n: NEXT n
 130 DATA 49,81,87,51,69,52,82,8
4,54,89,55,85,56,73,79,48,80
 140 FOR n=0 TO 16: READ m: POKE
 61000+n,m: NEXT n
 150 DATA 1,3,6,7,9,10,12,15,16,
18,19,21,22,24,27,28,30
 160 DIM a$(32): LET pam=50000:
LET tab=60000: LET poz=61000: PO
KE 23658,8
 170 BRIGHT 1
 180 FOR n=11 TO 15:
                    PRINT AT n,0;"
▀  ▀ ▀   ▀ ▀ ▀  ▀
": NEXT n
 190 PRINT AT 16,0;                "
 1  3 4   6 7 8   0
"
 195 PRINT AT 17,0;   a$;a$;a$;"
   Q  W  E  R  T  Y  U  I  O  P
";a$
 200 FOR n=16 TO 232 STEP 24: PL
OT n,0: DRAW 0,86: NEXT n
 210 GO SUB 390
 220 BRIGHT 0
 230 PAUSE 0: PRINT AT 2,10; BRI
GHT 1;" REJESTRACJA ": PRINT #0;
AT 0,0;a$;"                X = ODTWARZA
NIE
 240 FOR n=1 TO 9999: PRINT AT 9
,15; BRIGHT 1;n
 250 LET a=250
 260 IF INKEY$="X" THEN GO TO 31
0
 270 IF INKEY$<>"" AND PEEK (COD
E INKEY$+tab)<>250 THEN LET t=CO
DE INKEY$: LET a=PEEK (tab+t)
 280 IF a<250 THEN PRINT AT 10,P
EEK (poz+a); BRIGHT 1;CHR$ 144;
BEEP .2,a-2: PRINT AT 10,PEEK (p
oz+a);" "
 290 IF a=250 THEN PRINT AT 8,15
;"  "
 300 POKE pam+n,a: NEXT n
 310 PRINT AT 2,10; BRIGHT 1; IN
VERSE 1;" ODTWARZANIE ": BORDER
4: GO SUB 400: FOR m=1 TO n-1: L
ET p=PEEK (pam+m): PRINT AT 8,15
; BRIGHT 1; INVERSE 1;m+1;
 320 IF p<250 THEN PRINT AT 10,P
EEK (poz+p); BRIGHT 1; INVERSE 1
;CHR$ 144: BEEP .2,p-2: PRINT AT
 10,PEEK (poz+p);" "
 330 IF p=250 THEN PAUSE 1
 340 NEXT m
 350 PRINT #0;AT 0,0;a$;"
POWTORZYC ( T/N ) ?       "
 360 PAUSE 0: IF INKEY$="T" THEN
 PRINT AT 8,15;"          ": GO SUB 40
0: GO TO 310
 370 BORDER 1: FOR n=2 TO 9: PRI
NT AT n,0;a$: NEXT n: GO SUB 400
: GO SUB 390: GO TO 230
 390 PLOT 0,0: DRAW 255,0: DRAW
0,175: DRAW -255,0: DRAW 0,-175:
 PLOT 0,87: DRAW 255,0: RETURN
 400 PRINT #0;AT 0,0;a$;a$: RETU
RN
 900 CLEAR : SAVE "KLAWIATURA" L
INE 70: BEEP .5,-10: CLS : PRINT
 '"Verify: KLAWIATURA": VERIFY
"": RESTORE : GO TO 70
```

J. Rubinowicz

Spis najciekawszych programów

Poniżej przedstawiamy spis najciekawszych programów narzędziowych jakie ukazały się w IKS-ie, BAJTKU, KOMPUTERZE i MIKROKLANIE (od pierwszych numerów) na SPEKTRUM, AMSTRADA, COMMODORE i ATARI.
W tabeli podano nazwę prezentowanego programu, jego funkcję i źródło tj.numer pisma (dla IKS-a literka 'Z' oznacza - ZESZYT PROGRAMÓW)/rok ukazania się/skrót nazwy pisma, gdzie:

IKS - INFORMATYKA,KOMPUTERY SYSTEMY,
BAJ - BAJTEK,
KOM - KOMPUTER,
MIK - MIKROKLAN.

Lp	NAZWA PROGRAMU/FUNKCJA - JĘZYK PROGRAMU	ZRÓDŁO
1	2	3
	Z X - S P E K T R U M	
1	2	3
1	/Przenoszenie programów (zbiorów danych) ze SPEKTRUM na AMSTRADA - BASIC	Z2/87/IKS
2	ASEMBLER/Symulacja prawdziwego assemblera na SPEKTRUM - ASSEMBLER	Z2/87/IKS
3	KATALOG/Katalogowanie gier - BASIC	Z4/87/IKS
4	COMPACT/Zmniejsza zajętość pamięci,zwiększa szybkość działania programów -BASIC	2/86/IKS
5	/Zapamiętywanie 100 obrazów w 48k RAM-u SPEKTRUM - BASIC	5/86/IKS
6	/Zwiększanie tablicy zmiennych - BASIC	6/86/IKS
7	/Edytor znaków - BASIC	2/87/IKS
8	/Test monitora - BASIC	4/87/IKS
9	/Zwiększanie napisów na ekranie - BASIC	2/86/BAJ
10	/64 kolumny tekstu na ekranie - BASIC	7/86/BAJ
11	KATALOG/Spis programów na taśmie - BASIC	10/86/BAJ
12	/Renumeracja linii programowych - BASIC	11/86/BAJ
13	/Rozszerzanie tablic w pamięci - BASIC	4/87/BAJ
14	/śledzenie wykonywania programu - BASIC	11/87/BAJ
15	/Wyszukiwanie i tworzenie tablic występowania zmiennych w programie - BASIC	3/87/KOM
16	/Nagrywanie programu w dowolnym kodzie, umieszczanie pod wskazanym adresem-BASIC	3/87/KOM
17	/Sortowanie tablicy znakowej dwuwymiarowej - BASIC, - ASSEMBLER	5/87/KOM
18	/Tworzenie linii DATA z kolejnymi bajtami programu maszynowego - BASIC	12/87/KOM
	A M S T R A D	
1	2	3
1	/Przenoszenie programów (zbiorów danych) ze SPEKTRUM na AMSTRADA - BASIC	Z2/87/IKS
2	/Kopiowanie ekranu monitora - BASIC	4/87/IKS
3	/Zniesienie protekcji odczytu - BASIC	6/87/IKS
4	/Odtwarza skasowany plik na dysku -BASIC	7/86/BAJ
5	/Katalog dysku - BASIC	8/86/BAJ
6	/Analiza nazw zmiennych - BASIC	8/86/BAJ
7	/Zdejmowanie zabezpieczenia programów napisanych w BASIC-u - BASIC	9/86/BAJ
8	MONITOR/Przeglądanie i modyfikacja pamięci - BASIC	11/86/BAJ
9	/Emulator BASIC-a 1.1 dla CPC 464 -BASIC	9/87/BAJ 10/87/BAJ
	C O M M O D O R E	
1	2	3
1	DISASEMBLER/Tłumaczy instrukcje z kodu wewnętrznego na język symboliczny- BASIC	5/86/IKS
2	MANIAKTURBO/Usprawnienie interpretera	3-4/86BAJ
3	/Katalog dysku na C64 - BASIC	9/86/BAJ
4	TSLCOPY/Kopiowanie na dysk programow z pamięci komputera - BASIC	11/86/BAJ
5	LISTER/Zmiana szerokości wydruku	11/86/BAJ
6	/Łączenie programów w BASIC-u - BASIC	4/87/BAJ
7	TURBO16/Przyśpieszone ładowanie programów na C16 i C116 - BASIC	5/87/BAJ
8	AUTONUMBER/Automatyczne numerowanie linii programów w BASIC-u - BASIC	5/87/BAJ
9	/Połączenie C64 z drukarkami D-100 lub DZM-180 - BASIC	6/87/BAJ
10	/Przypisywanie rozkazów lub funkcji klawiszom funkcyjnym - BASIC	6/87/BAJ
11	/Katalog dysku	6/87/BAJ
12	/Przenoszenie obrazu z ekranu na drukarkę dla C16/116 PLUS/4	9/87/BAJ
13	DOS+/Rozszerza funkcje DOS-u (C64)-BASIC	10/87/BAJ
14	DISK-PROTECTOR/Zabezpiecza pliki przed skasowaniem - BASIC	11/87/BAJ
15	/Usprawnienie programu TURBO16 - BASIC	11/87/BAJ
16	/Obsługa dysku MS-DOS dla C128 - BASIC	9/87/KOM
17	TURBOTAPE64/Przyśpiesza operacje zapisu, odczytu i weryfikacji programów na taśmę magnetyczną dla C64 - BASIC	2/86/MIK
	A T A R I	
1	2	3
1	/Miniassembler - BASIC	Z1/87/IKS
2	/Automatyczny start programów - BASIC	Z2/87/IKS
3	DYSKMENU/Przeglądanie dyskietki i automatyczne uruchamianie programu - BASIC	Z2/87/IKS
4	/Symulacja funkcji assemblera - BASIC	Z4/87/IKS
5	/Konwerter języka maszynowega - BASIC	Z4/87/IKS
6	DISASEMBLER/Tłumaczy instrukcje z kodu wewnętrznego na język symboliczny -BASIC	Z4/87/IKS
7	/Edytor - BASIC	4/87/IKS
8	MINI-DOS/Realizuje funkcje DOS-u - BASIC	6/87/IKS
9	/Kopiowanie programów napisanych w języku wewnętrznym - BASIC	6/87/IKS
10	/Tworzenie procedur wyświetlania - BASIC	7-8/87IKS
11	/Kopiowanie programów - BASIC	7-8/87IKS
12	/Test stacji dysków - BASIC	7-8/87IKS
13	DISASEMBLER/Tłumaczy instrukcje z kodu wewnętrznego na język symboliczny -BASIC	10/87/IKS
14	SUMY KONTROLNE/Ułatwia wprowadzanie, drukowanie programów - BASIC	11/87/IKS
15	/Wyświetla na ekranie mapę dysku - BASIC	11/87/IKS
16	/Edytor kodu maszynowego - BASIC	12/87/IKS
17	RENUMERATOR/Renumeracja linii programowych programu - BASIC	5-6/86BAJ
18	TAPECOPIER/Kopiuje programy taśmowe	11/86/BAJ
19	SYMULATOR 6502/Analiza działania programu w kodzie maszynowym - BASIC	2/87/BAJ
20	/Katalog dysku, usuwanie plików - BASIC	2/87/BAJ
21	/Kasetowy system operacyjny - BASIC	5/87/BAJ
22	SPEED TRANS/Przemieszczanie danych w pamięci komputera - BASIC	8/87/BAJ
23	/Dynamiczne przeglądanie pamięci - BASIC	8/87/BAJ
24	/Przenosi zawartości zbiorów dyskowych "do wnętrza" instrukcji DATA - BASIC	2/87/KOM

INFORMATYCZNY SŁOWNIK ANGIELSKO-POLSKI

DIGITAL INFORMATION DISPLAY - cyfrowy monitor ekranowy,
DIGITAL INPUT - wejście cyfrowe,
DIGITAL INTEGRATED CIRCUIT - cyfrowy układ scalony,
DIGITAL OPTICAL RECORDING - zapis optyczny cyfrowy,
DIGITAL OUTPUT - wyjście cyfrowe,
DIGITAL PLOTTER - ploter, pisak cyfrowy, urządzenie rysujące,
DIGITAL PRINTER - drukarka cyfrowa,
DIGITAL READ-OUT - odczyt cyfrowy,
DIGITAL RECORDING - zapis cyfrowy,
DIGITAL REPRESENTATION OF DATA - cyfrowe przedstawienie danych,
DIGITAL SORT - sortowanie metodą cyfrową,
DIGITAL-TO-ANALOG CONVERTER - konwerter cyfrowo-analogowy,
DIGITIZE - przekształcać na postać numeryczną, dyskretyzować,
DIGITIZER - przetwornik analogowo-cyfrowy,
DIGIT KEYBOARD - klawiatura cyfrowa,
DIGIT POINT - przecinek (np. dziesiętny),
DIGIT SAFEGUARDING CODE - kod zabezpieczający cyfry,
DIMENSION - wymiar,
DIMENSIONLESS NUMBER - liczba bezwymiarowa,
DIP SWITCH - zestaw przełącznikowy wzdłużny,
DIRECT - bezpośredni, także: kierować,
DIRECT ACCESS - dostęp bezpośredni,
DIRECT ACCESS DEVICE - urządzenie z bezpośrednim dostępie,
DIRECT-ACCESS STORAGE - pamięć o dostępie bezpośrednim,
DIRECT ADDRESS - adres bezpośredni,
DIRECT ADDRESSING - adresowanie bezpośrednie (adres argumentu bezpośrednio za rozkazem),
DIRECT ARGUMENT - argument bezpośredni, występuje bezpośrednio po rozkazie,
DIRECT CODE - kod prosty,
DIRECT COMPUTER DISPLAY - ekranopis,
DIRECT CONTROL - sterowanie bezpośrednie,
DIRECT DIGITAL CONTROL - sterowanie cyfrowe bezpośrednie,
DIRECT DRIVE - synchronizacja bezpośrednia,
DIRECT-DRIVE MONOCHROME MONITOR - monitor monochromatyczny o bezpośredniej synchronizacji,
DIRECTED BEAM - konturowe rozwinięcie (kreślenie) obrazu,
DIRECTED BEAM CRT DISPLAY DEVICE - monitor konturowy,
DIRECTED BEAM DEVICE - konturowe urządzenie graficzne,
DIRECTED BEAM DISPLAY DEVICE - patrz: DIRECTED BEAM DEVICE,
DIRECT INFORMATION - informacja adresowana, informacja indywidualna,
DIRECT INPUT-OUTPUT - bezpośrednie przesyłanie informacji między mikroprocesorem a urządzeniami zewnętrznymi,

DIRECT JUMP - skok bezpośredni, także: rozkaz skoku bezpośredniego,
DIRECT LINE - patrz: IMMEDIATE LINE,
DIRECT MEMORY ACCESS CONTROLLER - sterownik DMA (bezpośredniego dostępu do pamięci),
DIRECT MEMORY ACCESS - bezpośredni dostęp do pamięci,
DIRECT MODE - patrz: IMMEDIATE MODE,
DIRECT NUMERICAL CONTROL (DNC) - bezpośrednie sterowanie numeryczne,
DIRECT SUBTRACT COUNTER - licznik odejmujący,
DIRECT (DATA) TRANSMISSION - transmisja (danych) bezpośrednia,
DIRECTED SET - zbiór skierowany,
DIRECTION - kierunek (zwiększanie lub malenie adresów),
DIRECTORY - informator, katalog, kartoteka, skorowidz
DISABLE - wyłączyć z działania, unieruchomić,
DISABLE INTERRUPTS - wyłącz przerwania,
DISABLE PULSE - impuls blokujący, impuls zamykający,
DISADVANTAGE - wada, ujemna strona,
DISASSEMBLE - patrz: DISADVANTAGE,
DISCARD - odrzucić, zaniechać,
DISCONNECT - odłączać, rozłączać,
DISCONTINUE - przerywać,
DISCONTINUITY - brak ciągłości, przerwa,
DISCONTINUOUS - przerywany, nieciągły,
DISCREPANCY - rozbieżność, różnica (np. pomiarów),
DISCRETE - dyskretny, nieciągły, odrębny, pojedyńczy,
DISCRETE CHANNEL - kanał dyskretny, kanał ziarnisty,
DISCRETE INFORMATION SOURCE - dyskretne źródło informacji,
DISCRETE REPRESENTATION OF DATA - dyskretne przedstawienie danych,
DISCRIMINATE - rozróżniać,
DISCUSSION - analiza, dyskusja, omówienie (np. wyników),
DISJUNCTION - alternatywa,
DISK - dysk magnetyczny,
DISK AUXILIARY ROUTINE - program dyskowy pomocniczy,
DISK CARTRIDGE - pakiet, dysk wymienny,
DISK DRIVE - napęd dyskietkowy, napęd dyskowy,
DISKETTE - dyskietka, dysk elastyczny,
DISK FACE - strona dysku,
DISK LAYOUT - rozplanowanie układu dysku,
DISK MEMORY - pamięć (magnetyczna) dyskowa,
DISK OPERATING SYSTEM - system operacyjny dyskowy,
DISK PACK - patrz: DISK CARTRIDGE,
DISK PARAMETER BLOCK - blok parametrów dyskowych,
DISK STORAGE - patrz: DISK MEMORY,
DISK STORAGE DUMP - wydruk zawartości dysku, wypis zawartości dysku,
DISK STORAGE EXTRACT - wyciąg z dysku,
DISK STORAGE PRINT-OUT - patrz: DISK STORAGE DUMP,
DISK TRACK - ścieżka dyskowa,
DISK TRANSFER AREA - obszar (bufor) transmisji dyskowych,
DISK UNIT - jednostka pamięci dyskowej,
DISPARITY - rozbieżność,
DISPATCHER - program rezydent,
DISPATCHING - koordynacja, także: służba dyspozytorska,
DISPERSED STORAGE - zapamiętywanie rozproszone,
DISPLACE - przemieszczać, przesuwać, umieścić,
DISPLACEMENT - przesunięcie, także: nazwa adresu używanego przy adresowaniu indeksowym,

DISPLAING - obrazowanie,
DISPLAY - zobrazowanie, przedstawienie, także: monitor obrazowy, przedstawiać, obrazować,
DISPLAY ADAPTER - adapter ekranu,
DISPLAY BACKGROUND - tło (podstawa) obrazu,
DISPLAY BUFFER - bufor obrazu, pamięć obrazu,
DISPLAY COMMAND - komenda graficzna,
DISPLAY CONSOLE - konsola graficzna,
DISPLAY CYCLE - cykl odnawiania, cykl odświeżania,
DISPLAY DATA - dane graficzne,
DISPLAY DEVICE - graficzne urządzenie wyjściowe,
DISPLAY ELEMENT - element obrazu (np. punkt, odcinek, znak alfanumeryczny, itp.),
DISPLAY ENTITY - obiekt graficzny (zbiór elementów obrazu, którymi można operować jako całością),
DISPLAY FILE - plik graficzny,
DISPLAY FOREGROUND - przedni plan obrazu,
DISPLAY FRAME - kadr obrazu,
DISPLAY GENERATOR - generator obrazu (przetwarza zawartość bufora obrazu na obraz fizyczny),
DISPLAY IMAGE - kod kadru,
DISPLAY INSTRUCTION - komenda graficzna,
DISPLAY ORDER - rozkaz graficzny,
DISPLAY PAGE - stronica obrazowa,
DISPLAY PANEL - panel graficzny, konsola graficzna oprogramowana w sposób umożliwiający pracę w trybie interakcyjnym,
DISPLAY PRIMITIVE - element obrazu (np. punkt, odcinek, znak alfanumeryczny, itp.),
DISPLAY SCOPE - wielkość obrazu, rozmiar rysunku,
DISPLAY SPACE - przestrzeń operacyjna urządzenia, (obszar, którego zawartość jest przedstawiona na powierzchni obrazowania),
DISPLAY SURFACE - powierzchnia obrazowania (nośnik, na którym tworzony jest obraz fizyczny),
DISPLAY UNIT - monitor ekranowy, ekranopis, także: wskaźnik radarowy,
DISPLAY WRITER - element kreślący obraz,
DISPOSE OF - rozporządzać (czymś), dysponować,
DISREGARD - ignorować, pomijać, nie zważać (na coś),
DISSEMINATE - rozpowszechniać, rozprowadzać (np. informacje),
DISSEMINATION - rozpowszechnianie, rozprowadzanie,
DISSIMILAR - niepodobny, różny,
DISTANCE CONTROL - sterowanie zdalne,
DISTINCTION - różnica,
DISTORT - wypaczyć, uszkodzić, zniekształcić,
DISTORTION - spaczenie, wypaczenie, wykrzywienie, zniekształcenie,
DISTORTION FACTOR - współczynnik zniekształcenia,
DISTRIBUTE - rozdzielać,
DISTRIBUTED PROCESSING - przetwarzanie rozproszone,
DISTRIBUTING BOARD - tablica rozdzielcza,
DISTRIBUTION - rozłożenie, rozkład, rozprowadzenie,
DISTURB SIGNAL - sygnał zakłócający,
DISTURBANCE - zakłócenie, zaburzenie,
DIVERGE - odchylać się, odbiegać,
DIVERGENCY - dywergencja, rozbieżność,
DIVIDE - dzielić, także: podziel,
DIVIDEND - dzielna,
DIVIDER - dzielnik,
DIVISION - dzielenie, znak dzielenia, także: w COBOLU: dział (sekcja),
DIVISION SIGN - znak dzielenia,
DIVISION WITH ROUND-OFF - dzielenie z zaokrągleniem,
DMAC - patrz: DIRECT MEMORY ACCESS CONTROLLER,

DML - patrz: DATA MANIPULATION LANGUAGE,
DNC - patrz: DIRECT NUMERICAL CONTROL,
DO - patrz: DIGITAL OUTPUT,
DOC - patrz: DIAGNOSTICS ON-CHIP,
DOCUMENT - dokument,
DOCUMENT-FREE DATA PROCESSING - przetwarzanie danych bezdokumentowe,
DOCUMENT PROCESSING - przetwarzanie dokumentów,
DOCUMENT READER - czytnik dokumentów,
DOCUMENT SAFEGUARDING - ochrona dokumentów,
DOCUMENTATION - dokumentacja,
DOMAIN - dziedzina, obszar,
DOMAIN OF MAP - dziedzina odwzorowania,
DOR - patrz: DIGITAL OPTICAL RECORDING,
DOT AND ELEMENT - pseudoelement "I",
DOT DIAGRAM - wykres punktowy,
DOT MATRIX - matryca punktów,
DOT MATRIX CHARACTER GENERATOR - generator mozaikowy znaków,
DOT MATRIX PRINTER - drukarka mozaikowa,
DOT MEMORY - pamięć punktowa,
DOT OR ELEMENT - pseudoelement "LUB",
DOT PRINTER - patrz: DOT MATRIX PRINTER,
DOT PRODUCT - iloczyn skalarny,
DOUBLE CURRENT - prąd dwukierunkowy,
DOUBLE DENSITY (DD) - podwójna gęstość, także: sposób zapisu na dysku,
DOUBLE-DENSITY FLOPPY DISK - dyskietka o podwójnej gęstości zapisu,
DOUBLE-LENGTH NUMBER - liczba o podwójnej długości,
DOUBLE PRECISION - podwójna precyzja,
DOUBLE-PRECISION COMPUTATION - liczenie z podwójną precyzją,
DOUBLE-PRECISION NOTATION - zapis podwójnej precyzji,
DOUBLE SIDED (DS) - dwustronny, także: zapis po obu stronach dysku,
DOUBLE-WIRE COMMUNICATION - przesyłanie dwuprzewodowe,
DOWNLOAD - ładowanie w dół, ładowanie na dolne pozycje (np. rejestru),
DOWNLOADING - ładowanie zdalne, ładowanie skrośne, ładowanie w dół,
DOWN-TIME - okres wyłączenia, okres niesprawności (urządzenia), czas przestoju, przestój spowodowany awarią sprzętu,
DOWN-TIME OPERATION - czynność związana z usunięciem uszkodzenia,
DOWN-UP COUNTER - licznik rewersywny (zliczający w obydwu kierunkach),
DPB - patrz: DISK PARAMETER BLOCK,
DPL - patrz: DESCRIPTOR PRIVILEGE LEVEL,
DRAGGING - wodzenie (np. przesuwanie elementu obrazu za piórem świetlnym),
DRAM - patrz: DYNAMIC RANDOM ACCESS MEMORY,
DRAVING AREA - wielkość (rozmiar) powierzchni obrazowania,
DRAW - rysować, także: wyciągać wniosek,
DRAWBACK - cecha ujemna, wada,
DRAWING SIZE - wielkość obrazu, rozmiar rysunku,
DRIVE - urządzenie, napęd,
DRIVER - program obsługi urządzenia, moduł sterujący,
DROP - opuszczać, także: spadek (np. napięcia),
DROP CABLE - kabel dołączeniowy,
DROP IN - sygnał nie zapisany (dot. zapisu na nośniku magnetycznym),

DROP OUT - sygnał nie odebrany (dot. zapisu na nośniku magnetycznym),
DROPPED BIT - bit opuszczony, bit zagubiony,
DRUM - bęben,
DRUM MEMORY - pamięć (magnetyczna) bębnowa,
DRUM PLOTTER - pisak bębnowy, ploter bębnowy,
DRUM PRINTER - drukarka bębnowa,
DRY RUN - przebieg próbny,
DS - patrz: DATA SEGMENT,
DT-EQUIPMENT (DTE) - urządzenie transmisji danych, UTD,
DTA - patrz: DISK TRANSFER AREA,
DTE - patrz: DT-EQUIPMENT,
DU - patrz: DISK UNIT,
DUALCODE - kod dwójkowy,
DUAL FULL ADDER - sumator pełny dwójkowy,
DUAL PAPER FEED - przesuw podwójny, posuw podwójny,
DUAL-PURPOSE KEY - klawisz dwuczynnościowy,
DUAL-TRACK RECORDIND - zapisywanie dwuścieżkowe,
DUE DATE - termin,
DUMB TERMINAL - "niemy terminal",
DUMMY - element fikcyjny, upozorowany, sztuczny,
DUMMY ARGUMENT - argument formalny,
DUMMY DEVICE - urządzenie pozorne,
DUMMY INDEX - niemy indeks,
DUMMY INSTRUCTION - rozkaz pusty,
DUMMY NAME - nazwa pusta,
DUMMY ORDER - rozkaz fikcyjny,
DUMMY PARAMMETER - parametr formalny,
DUMMY ROUTINE - ślepy (bezczynny) podprogram,
DUMMY SECTION - pseudosekcja,
DUMMY VARIABLE - zmienna fikcyjna,

DUMP - przeniesienie całego lub częściowego obszaru pamięci komputera do innego obszaru pamięci lub na nośnik zewnętrzny,
DUMP CHECK - sprawdzenie wydruku pamięci,
DUMPING - przenoszenie zawartości pamięci komputera, przekazywanie,
DUMPING PROGRAM - program post-mortem,
DUMPING SYSTEM CONTROL AREA - kopiowanie obszaru sterowania,
DUODECIMAL - dwunastkowy (system liczenia),
DUPLEX - transmisja jednoczesna, transmisja dupleksowa, duplęx,
DUPLEX CHANNEL - kanał dupleksowy,
DUPLEX OPERATION - patrz: DUPLEX,
DUPLICATE - kopia, także: powielać, reprodukować, powtarzać,
DUPLICATING PROGRAM - program duplikujący, program kopiujący,
DUPLICATING PUNCH - dziurkarka powtarzająca się, dziurkarka reprodukująca,
DUPLICATION - kopiowanie, powielanie,
DUPLICATION CHECK - kontrola przez duplikację,
DUPLICATION METHOD - metoda powielania,
DURABILITY - trwałość,
DURATION - czas trwania,
DUSTPROOF BAG - pyłoszczelna torebka (np. do przechowywania taśmy magnetycznej),
DUTY CYCLE - cykl obowiązkowy, cykl pracy,
DUTY PROGRAM - program usługowy, program użytkowy,
DX (DATA REGISTER) - rejestr danych,
DYADIC - dwuczłonowy, podwójny,
DYNAMIC IMAGE - przedni plan obrazu,
DYNAMIC MAIN STORAGE ALLOCATION - przydzielanie dynamiczne pamięci operacyjnej,

DYNAMIC MAPPING SYSTEM - system z odwzorowaniem dynamicznym,
DYNAMIC PROGRAM RELOCATION - przesunięcie dynamiczne programu,
DYNAMIC PROGRAMMING - programowanie dynamiczne,
DYNAMIC RANDOM ACCESS MEMORY - dynamiczna pamięć o dostępie bezpośrednim,
DYNAMIC SKEW - przekos dynamiczny,
DYNAMIC STORAGE - pamięć dynamiczna,
DYNAMIC TESTING TECHNIQUE - testowanie dynamiczne,

E

E^2 - patrz: ELECTRICALLY ERASABLE DEVICE,
EA - patrz: EFFECTIVE ADDRESS,
EAC - patrz: END-AROUND CARRY,
EARLY FAILURE PERIOD - okres uszkodzeń początkowych,
EARN - patrz: EUROPEAN ACADEMIC RESEARCH NETWORK,
EAROM - patrz: ELECTRICALLY ALTERABLE ROM,
EARTH - uziom, zwarcie doziemne, także: uziemiać,
EARTH BAR - szyna uziemiająca,
EARTH TERMINAL - zacisk uziomowy,
EBCDI-CODE - patrz: EXTENDED BINARY-CODED DECIMAL INTERCHANGE CODE,
EBCDIC - patrz: EXTENDED BINARY-CODED DECIMAL INTERCHANGE CODE,
EBNF - patrz: EXTENDED BACKUS-NAUR FORM,
ECC - patrz: ERROR CORRECTION CODE,
ECGB - patrz: ENHANCED COLOR GRAPHICS BOARD,
ECHO - echo, potwierdzenie poprawności odbioru wysłanej informacji,
ECHO-CHECK - kontrola zwrotna, echo kontrolne,
ECHO-PLEX - echo-pleks,
ECHO-PRINTING - drukowanie echa,
ECHO SUPPRESSOR - tłumik echa,
ECHOING THE CHARACTER - potwierdzenie poprawności odbioru wysłanego znaku,
ECMA - patrz: EUROPEAN COMPUTER MANUFACTURES' ASSOCIATION,
ECTL - patrz: EMITTER-COUPLED TRANSISTOR LOGIC,
EDC - patrz: ERROR DETECTION/CORRECTION,
EDGE - krawędź (np.karty, taśmy papierowej),
EDGE-NOTCHED CARD - karta (kartotekowa) selekcyjna nacinana,
EDGE PATH - droga krawędziowa, ścieżka krawędziowa,
EDGE-PUNCHED CARD - karta (kartotekowa) obrzeżnie dziurkowana,
EDGE-TRIGGERED WRITE - zapis przełączany krawędzią,
EDGEWISE - wzdłuż krawędzi, na kant,
EDIT - redagować, także: edytor,
EDIT-DIRECTED - redagowanie kierowane (sterowane),
EDITING - redagowanie wyników, przygotowywanie informacji,
EDITING KEY - klawisz edycji,
EDITING STATEMENT - dyrektywa edycji,
EDITING TERMINAL - terminal w skomputeryzowanym systemie wydawniczym,
EDITION - wydanie,
EDITOR - edytor, program redagujący,

PROGRAM 51 ATARI

Enigma

Za pomocą tego programu można kopiować gry i programy użytkowe do pojemności 33 kB pamięci. W zależności od stosowanego nośnika oryginału oraz jego wtórnika mamy do wyboru trzy możliwości kopiowania tj.: z taśmy na taśmę; z dysku na taśmę i z taśmy na dysk. Obsługa programu jest bardzo prosta i polega na wykonywaniu poleceń zobrazowanych na ekranie monitora. Program składa się z trzech części. Pierwsza to zwiastun, po wprowadzeniu którego komputer oczekuje na wpisanie z klawiatury instrukcji RUN. Instrukcja ta uruchamiając zwiastuna umożliwia wprowadzenie części drugiej (objaśnienie) oraz programu głównego w formie gotowej do wykonywania kopii. Do sterowania programem wykorzystywane są cztery klawisze:
T — taśma lub tak w zależności od pytania;
D — dysk; N — nie i SPACE do potwierdzenia wyboru.

H. JANKOWSKI

```
VH 0 POKE 16,64:POKE 53774,64:POKE
   709,14:POKE 710,0:POKE 752,1
FB 1 ? CHR$(125):POSITION 10,19:? "
   Fanom ATARI-Redakcja IKS-a":FOR
   I=1572 TO 1600:READ A:POKE I,A:N
   EXT I:U=USR(1572)
MH 2 DATA 169,12,141,252,2,169,1,14
   1,68,2,162,253,154,169,183,72,16
   9,84,72,169,4,32,182,187,169,255
   ,76,4,187

WX 3 GRAPHICS 0:POKE 16,64:POKE 537
   74,64:POKE 710,0:POKE 752,1:POKE
    40002,8:DL=PEEK(560)+256*PEEK(5
   61):POKE DL+5,7
ZP 4 FOR I=14 TO 20:POKE DL+I,6:NEX
   T I:POKE 709,14:POKE 82,0:? "
    ***************************"
WZ 5 ? "    atari-enigma
   ****************(X-1987)**":POSI
   TION 24,4:? "- kopiuje programy
   o pojemnosci ";
KJ 6 POSITION 34,5:? "do 33 kBajt,"
   :POSITION 24,6:? "w/g jednej z w
   ybranych relacji :":? :? "    ta
   sma  tasma"
VJ 7 ? "    tasma   dysk":? "    dy
   sk   tasma":? :? "      Procesor
    MOSTEK 6502C (HALT)":POSITION 2
   4,16
JF 8 ? "POWODZENIA !":FOR I=1536 TO
    1571:READ A:POKE I,A:NEXT I:FOR
    I=1601 TO 1607:READ A:POKE I,A:
   NEXT I:I=USR(1572)
KG 9 DATA 104,104,104,162,16,157,66
   ,3,104,157,69,3,104,157,68,3,104
   ,157,73,3,104,157,72,3
JP 10 DATA 32,86,228,132,212,169,0,
   133,213,133,77,96
NC 11 DATA 104,169,0,141,65,6,96
SS 20 CLR :OPEN #2,4,0,"K:":OPEN #3
   ,8,0,"E:"
AE 25 POKE 16,64:POKE 53774,64:POKE
    82,2:POKE 709,14:POKE 710,0
NG 30 DIM F$(16),FR$(16),FL$(6),AD(
   9),ST(9)
FX 40 AP=PEEK(14)+256*PEEK(15)
SX 50 DL=PEEK(560)+256*PEEK(561).
PZ 60 ST=AP+54:L=DL-ST-6
RU 70 U=USR(1601)
ZK 90 ? "}"
ZC 91 LN=90:OP=1:?
GY 95 LZ=0:AD(0)=0:ST(0)=ST:KOP=0
UU 100 ? "Pochodzenie programu: ";C
   HR$(212);"asma - ";CHR$(196);"ys
   k ? ";
LF 110 GET #2,AN:IF AN<>68 AND AN<>
   84 THEN 110
XZ 115 DYSK=0:PUT #3,AN:?
MK 120 IF AN=68 THEN DYSK=1
DC 125 IF  NOT DYSK THEN F$="C:":DY
   R=128:GOTO 160
UV 127 GOSUB 900
YM 130 ? :? "Nazwa programu: ";:INP
   UT F$
II 135 IF F$="" THEN ? CHR$(28);CHR
   $(156);CHR$(28);:GOTO 130
RB 140 GOSUB 2000
DN 150 DYR=0
WW 160 RT=160:K=0:TRAP 5000
CY 162 KOP=KOP+1:ST(KOP)=ST(KOP-1)+
   AD(KOP-1)+10:LZ=LZ+AD(KOP-1)+10
XA 170 OPEN #1,4,DYR,F$
QJ 175 X=USR(1536,7,ST(KOP),L-LZ)
MJ 178 CLOSE #1
JG 180 TRAP 40000
PI 185 IF X=1 OR X=136 THEN 240
UO 190 IF DYSK THEN ? :? "Stacja dy
   skowa";
SE 200 IF  NOT DYSK THEN ? :? "Magn
   etofon";
JL 210 ? " nie pracuje."
YU 215 RT=160:K=1:KOP=KOP-1:GOTO 51
   0
PR 240 IF X=1 THEN ? :? "Zbyt dlugi
    program !":IF KOP>1 THEN KOP=KO
   P-1.00000085:GOTO 3000
OX 241 IF X=1 AND KOP=1 THEN GOTO 9
   1
QP 250 AD=PEEK(856)+256*PEEK(857)
RJ 251 AD(KOP)=AD
MC 253 ? :? "Pojemnosc ";KOP;" czes
   ci - ";AD;" bajt":IF DYSK THEN 2
   65

PM 258 IF KOP=9 THEN 3000
XP 260 ? "Czy wgrywac dalej ? ";
EA 261 GET #2,AN:IF AN=78 AND KOP>1
    THEN ? :GOTO 3000
GA 262 IF AN=84 THEN ? "Tak":GOTO 1
   60
YY 263 IF AN=78 AND KOP=1 THEN ? "N
   ie":GOTO 265
QB 264 GOTO 262
RO 265 D=DYSK
JR 268 LN=268:OP=0:DYSK=0
CR 270 ? :? "Wgrac na ";CHR$(212);"
   asme czy ";CHR$(196);"ysk ? ";
FV 275 GET #2,AN:IF AN<>68 AND AN<>
   84 THEN 275
XZ 280 DYSK=0:PUT #3,AN:?
MZ 290 IF AN=68 THEN DYSK=1
DV 295 IF  NOT DYSK THEN DYR=128:F$
   ="C:":GOTO 360
EQ 300 DYR=0:GOSUB 900
OG 302 IF  NOT D THEN 330
IY 303 ? "Poprzednia nazwa: ";F$
EB 305 ? :? "Czy zmieniasz nazwe ?
   ";
PH 310 GET #2,AN:IF AN<>78 AND AN<>
   84 THEN 310
MH 320 ? :IF AN=78 THEN 360
US 330 ? :? "Nowa nazwa: ";:INPUT F
   $
MQ 335 IF F$="" THEN ? CHR$(28);CHR
   $(156);CHR$(28);:GOTO 330
RF 350 GOSUB 2000
FX 360 I=1
SZ 400 GOSUB 7000
MT 420 ? :? "Skopiowano !"
LM 430 ? :? "Czy kopiowac dalej ? "
VZ 440 GET #2,AN:IF AN<>78 AND AN<>
   84 THEN 440
UP 450 IF AN=84 AND KOP=1 THEN 268
NS 455 IF AN=84 AND KOP>1 THEN 3000
ST 460 GOTO 90
UH 500 ? :? "Pomylka ";PEEK(195)
MJ 510 POKE 732,0:POKE 764,255:POKE
    53279,7:POKE 752,1:?
QZ 511 IF (KOP>1 OR K) AND OP THEN
   ? "[OPTION] zachowuje sie poprze
   dnia czesc"
OT 512 ? "[SELECT] powtorz ostatni
   I/O proces"
OP 513 ? "[START] powrot do ATARI-E
   NIGMA"
NV 514 ? "[HELP]  powtorz ostatnie
   I/O dzialanie"
FR 520 IF PEEK(732)=17 THEN POKE 75
   2,1:GOTO LN
BE 522 IF PEEK(53279)=6 THEN POKE 7
   52,1:GOTO 90
WF 523 IF (KOP>1 OR K) AND OP AND P
   EEK(53279)=3 THEN POKE 752,1:GOT
   O 3000
ZA 525 IF PEEK(53279)=5 THEN POKE 7
   52,1:GOTO RT

UE 530 POKE 764,255:GOTO 520
TP 900 ? :? "Podaj numer stacji: ";
ZS 910 GET #2,AN:DRN=AN-48:IF DRN<1
    OR DRN>4 THEN 910
LK 915 PUT #3,AN:?
IW 920 FR$="D":IF DRN>1 THEN FR$(LE
   N(FR$)+1)=STR$(DRN)
ZI 925 FR$(LEN(FR$)+1)=":":FL$=FR$:
   FL$(LEN(FL$)+1)="*.*"
QQ 930 ? :? "Czytnik ";DRN;" szuka
   w spisie ":?
QL 935 RT=935:TRAP 500
LG 940 CLOSE #1:OPEN #1,6,0,FL$:CLO
   SE #1
JV 945 TRAP 40000
ZP 950 RETURN
UB 2000 F=0
MB 2010 FOR I=1 TO LEN(F$)
SM 2015 F=F+(F$(I,I)=":")
EW 2020 NEXT I
GM 2025 IF F=0 THEN FR$(LEN(FR$)+1)
   =F$:F$=FR$
AJ 2030 RETURN
FK 3000 POKE 201,5:LN=3000:OP=0
PJ 3005 ? :? CHR$(17);:FOR A0=1 TO
   22:? CHR$(18);:NEXT A0:? CHR$(5)
VU 3010 FOR I=1 TO KOP
GR 3020 ? ,"| Czesc ";I;" - ";AD(I)
   ;
PV 3030 POKE 85,24:? " bajt !"
FD 3040 NEXT I
UC 3050 ? ,CHR$(26);:FOR A0=1 TO 22
   :? CHR$(18);:NEXT A0:? CHR$(3)
CY 3060 ? :? "Wgrywam !":?
AU 3070 F$="C:":DYR=128
QA 3075 POKE 752,1
WP 3080 FOR I=1 TO KOP
HQ 3090 ? I;" ";
ZH 3100 GOSUB 7000
EW 3110 NEXT I
OH 3115 POKE 752,1:?
PH 3120 GOTO 420
CK 5000 KOP=KOP-1:GOTO 500
VP 7000 RT=7000:TRAP 500
LN 7010 IF I>1 THEN POKE 764,12
SH 7020 OPEN #1,8,DYR,F$
YV 7030 X=USR(1536,11,ST(I),AD(I))
NB 7040 CLOSE #1
OQ 7050 TRAP 40000
LV 7060 IF X=1 THEN 7090
TG 7070 IF DYSK THEN ? "Pomylka ";X
SN 7080 IF  NOT DYSK THEN ? "Pomylk
   a ";X
EE 7085 RT=7090:GOTO 510
BG 7090 RETURN
```

KRZYŻÓWKA NR 4

	1	2	3	4	5	6	7
8	R	E	Ż	Y	S	E	R
	O		O		U	9 O	K
10	G	O	T	O	W	A5	S
	Ó		A7		N	11 Z1	A
12	W	E26	D8	L	I13	N	A
	K	13 E		14 C18		15 D21	A
16	A	N	K	I	E17	T	A
		U		R			21 L
22	A	M4	B	A	S24	A	D
		I		K			23
24		Z22		L			
		M6		I25			25 A
26		A3		O10		27	
		T		N		32	27 A

	5	6	7
8	D	S2	W
	R31	U T N	16 K
10	A	O	E
	Ż Y Ł O	Ś11	Ć
12	E	Ó	N
	A T O W	N15	I23 K
16	K	K	A 20
	A30	T A R	K A
22	A	R	A14 19
		A28	I13
24		K	
	N T	O20 N	O W
26		A	
	U	T29 O	K A R9

POZIOMO: 8) Krzysztof Zanussi, 9) był nim Neron, 10) niedoświadczony młokos, młodzik, 11) komitywa, poufałość, 12) salami albo kabanos, 15) przyrząd używany w kasach biletowych lub na poczcie, 16) badanie statyst. służące do obserwacji statystycznej, 21) podręczne źródło światła, 22) placówka dyplomatyczna, 23) przeciwdziałanie, 24) Fiodor (1873—1938), rosyjski śpiewak światowej sławy, bas, 25) radziecki konstruktor lotniczy, twórca m.in. samolotu AN-22, 26) część ringu, 27) turystyczny środek lokomocji,

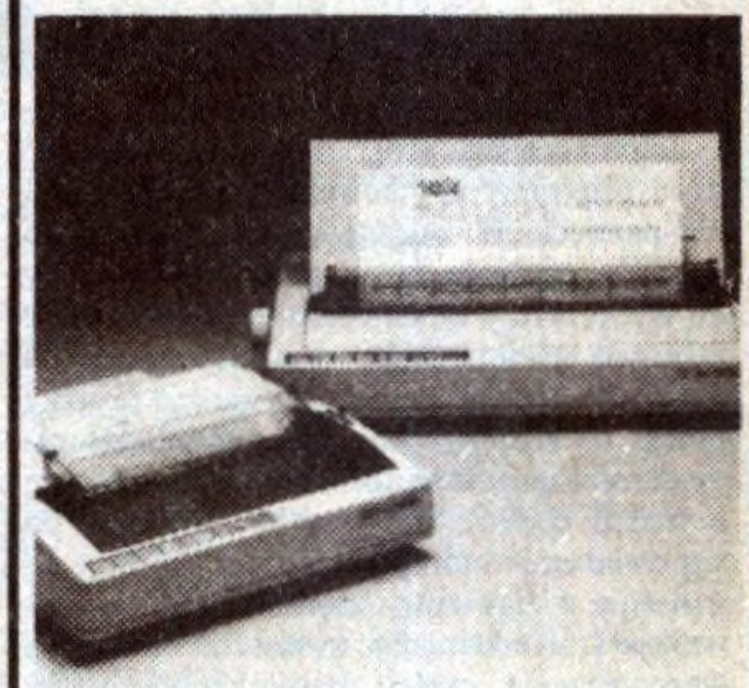
PIONOWO: 1) zewnętrzna, przezroczysta błona gałki ocznej, 2) między przełykiem a jelitem, 3) uniwersalne urządzenia transportowe, dźwignice, 4) manewr szachowy, 5) tabletka, 6) zakład zbiorowego żywienia, 7) chłop, 13) dawna moneta lub medal, mająca wartość zabytkową, 14) największe miasto i port Krety, 15) morska żegluga nieregularna, 17) port morski we wsch. części Cypru, 18) rozprawa naukowa, 19) bylina oleista uprawiana na obszarach tropikalnych dla nasion i do wyrobu oleju rycynowego, 20) Żyd Wieczny Tułacz, ze średniowiecznej legendy.

Litery z kratek ponumerowanych dodatkowo w prawym dolnym rogu od 1 do 32 utworzą hasło, które wystarczy nadesłać jako rozwiązanie zadania na kartkach pocztowych pod adresem redakcji, do końca czerwca, **naklejając kupon „IKS"-a**. Wśród autorów prawidłowych odpowiedzi rozlosujemy bony pieniężne i nagrody książkowe.

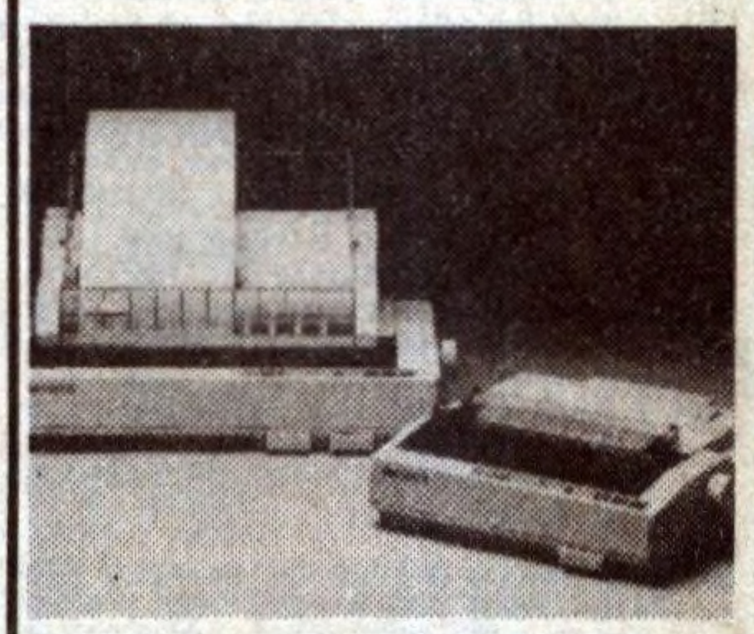
KUPON ▶

MIKROCIEKAWOSTKI

star



NR-10 i NR-15



NB 24-10 i NB 24-15

ND-10 i ND-15

„IKS" — dodatek „Żołnierza Wolności". Redaguje Wiesław Cetera (kierownik zespołu); Rada programowa: Krzysztof Chmarra, Romuald Głąb, Włodzimierz Gogołek, Janusz Janiec, Henryk Krasuski, Ireneusz Miernik, Ludwik Piela, Jacek Szaniawski. Adres redakcji: 00-950 Warszawa ul. Grzybowska 77, telefon centrali 20-12-61 w. 486. Telex 313664. Rękopisów nie zamówionych redakcja nie zwraca i zastrzega sobie prawo do skrótów. Nakładem: Wydawnictwa „Czasopisma Wojskowe", Warszawa ul. Grzybowska 77. Fotoskład i druk offsetowy — Wojskowe Zakłady Graficzne im. gen. dyw. A. Zawadzkiego. Nr zam. 1117. Nr ind. 361682. U-40