NR INDEKSU 353965
PL ISSN 0860-1674

Z MIKROKOMPUTEREM NA TY

9-10

Bajtek

MAGAZYN KOMPUTEROWY NR (55-56)'90 CENA 5000

CO Z OBROTÓW DYSKÓW WYNIKA DLA ZWYKŁEGO UŻYTKOWNIKA?

MS WINDOWS

BELFER Z DYSKIETKĄ

AMIGA 3000

TOTO-LOTEK

POLSKIE LITERY W Action!

AY i GRY

URIDIUM

DYLEMATY WZROSTU

W połowie września odbyła się w Błażejewku koło Poznania VI Konferencja „Informatyka w szkole". Wśród zaproszonych gości przeważali nauczyciele szkół średnich, w dziedzinie sprzętu królował standard IBM PC. Głównym tematem Konferencji była prezentacja oprogramowania dydaktycznego wytworzonego w kraju i przeznaczonego dla szkół podstawowych i średnich. Więcej informacji o wrażeniach z Błażejewka znajdą Państwo w artykule — „Belfer z dyskietką".

Posiadaczy ZX Spectrum zainteresowanych rozbudową swojego komputera odsyłamy do materiału opisującego rozszerzenie 80 KB. Z kolei osobom, które w domu dysponują sprzętem 8-bitowym, a w pracy komputerami IBM PC, polecamy artykuł w Klanie Amstrad poświęcony strukturze dyskietki systemu MS DOS-a. Jest to w dalszym ciągu dominujący system operacyjny komputerów typu IBM PC.

Wprowadzenie na szerszą skalę nowszego systemu OS/2 napotyka na przeszkody głównie natury sprzętowej — olbrzymie wymagania co do pamięci operacyjnej i dyskowej oraz szybkości zastosowanego procesora. Praktycznie korzystanie z OS/2 staje się możliwe na sprzęcie klasy AT 386 z pamięcią RAM 6MB i dyskiem twardym 120 MB. Na razie nie wszystkich na to stać. Etapem przejściowym może okazać się program-nakładka o nazwie MS WINDOWS, który ma wymagania znacznie skromniejsze. Informacje, czym naprawdę są wspomniane „okienka" można znaleźć w klanie IBM.

Ostatnio redakcja weszła w posiadanie m.in. komputera Amiga 500. Mamy nadzieję, że zakup ten zadowoli nie tylko właścicieli sprzętu firmy Commodore. Liczymy także na lepsze zdjęcia ekranów opisywanych na łamach „Bajtka" gier. O Amidze 500 pisaliśmy wcześniej, natomiast w tym numerze przedstawiamy krótki opis najnowszej, 32-bitowej wersji 3000.

W tym miejscu pora na pewną refleksję. Większość komputerów w polskich domach to jeszcze sprzęt 8-bitowy, praktycznie już nie produkowany. Z kolei w pracy coraz częściej ocieramy się o sprzęt 32-bitowy, a prywatnymi kanałami dociera do nas coraz więcej 16-bitowych Atari ST i wspomnianych wcześniej komputerów Amiga.

Jako pismo praktycznie od początku obecne na naszym rynku komputerowym nie możemy ignorować tych faktów. Nie możemy zatrzymać się w miejscu i obojętnie patrzeć, jak przestają nas czytać kolejne setki, czy tysiące posiadaczy nowszych komputerów, którzy wyrośli z „Bajtka". Dotychczasową reakcją na te zjawiska jest zwiększenie objętości pisma i wprowadzenie klanu IBM.

Nie zostawiamy jednak na pastwę losu właścicieli sprzętu 8-bitowego, obiecujemy im tę samą ilość materiałów o ich sprzęcie, ale dalsze „grubienie" „Bajtka" pójdzie głównie na opis komputerów 16-bitowych.

Hołdując zasadzie „dla każdego coś miłego", chcemy złagodzić przejście między tymi dwoma generacjami sprzętu.

Razem z drugim, wydanym samodzielnie numerem „Bajtka", wprowadziliśmy na rynek nowe czasopismo — „Moje Atari". Jest to dwumiesięcznik poświęcony komputerom firmy Atari, ale do końca tego roku będzie się on ukazywał co miesiąc. W numerze październikowym przeważają materiały poświęcone „małemu" Atari, ale mamy nadzieję, że z następnych wydań będą zadowoleni także właściciele Atari ST.

Trzeciemu zeszytowi Bajtka towarzyszy pierwszy numer magazynu gier komputerowych — „Top Secret". Podobnie jak „Moje Atari", jest to dwumiesięcznik, którego do końca roku ukażą się trzy numery. Mamy nadzieję, że nie będzie to tylko pismo dla dzieci i młodzieży — znajdą się tam też opisy trudniejszych, strategicznych gier, również na komputery IBM PC.

Wydawanie przez nas nowych pism nie oznacza, że rezygnujemy w samym „Bajtku" z prezentowanej, na ich łamach, tematyki. W dalszym ciągu „Bajtek" pozostaje naszym „oczkiem w głowie".

Jarosław Młodzki

Bajtek
MAGAZYN KOMPUTEROWY

Redaguje Kolegium:
Marcin Borkowski, Marek Czarkowski, Łukasz Czekajewski, Janusz Jarmoch, Jarosław Młodzki (red. nacz.), Waldemar Nowak, Maciej Pietraś, Marcin Przasnyski, Wanda Roszkowska, Wojciech Zientara
Andrzej Pilaszek
Mieczysław Płacheta
Marek Sawicki
Piotr Sumara
Michał Szokoło
Tomasz Tarczyński
Stanisław Winiecki

Szefowie Klanów:
Amstrad — Jonasz Mayer
Atari — Wojciech Zientara
Commodore — Klaudiusz Dybowski
Co Jest Grane — Łukasz Czekajewski
IBM — Marcin Borkowski
Micro Magazyn — Janusz Jarmoch
Spectrum — Maciej Pietraś

Stali współpracownicy:
Marcin Bójko
Jarosław Burczyński
Piotr Kos
Robert Magdziak
Grzegorz Ostapiuk

Opracowanie graficzne:
Wanda Roszkowska

Redaktor techniczny: Maria Radzimińska
Zdjęcia: Leopold Dzikowski

Wydawca:
Spółdzielnia „Bajtek"
ul. Wspólna 61
00-687 Warszawa

Skład i Druk:
Prasowe Zakłady Graficzne w Ciechanowie
Fotoskład: Grażyna Kurzątkowska
Montaż: Grażyna Ostaszewska
Korekta: Maria Krajewska
Teresa Rutkowska
Nakład: 105 tys. egz. Zam. 66820

Redakcja:
ul. Wspólna 61
00-687 Warszawa
Tel. 21-12-05

BELFER Z DYSKIETKĄ ... 3
MICRO MAGAZYN 4
DRUKOWANIE W SZPALTACH 7
KLAN COMMODORE 8
:Amiga 3000
:Mały Pomocnik
:Klub C128
:Tylko dla dyskomanów
Język "C" dla najmłodszych . 11
KLAN ATARI 12
:Mini edytor duszków
:Toto-lotek
:Sound Machine
:Polskie litery w Action!
:ADRES i DIR w Kyan Pascalu
KLAN SPECTRUM 15
:ZX Spectrum 80K
:Timex bez tajemnic cz. 3
:Interface monitora
:Język maszynowy cz. 2
:TOS inaczej
CO JEST GRANE 20
:Uridium
:Lista Przebojów
:SOS
:Forgotten Worlds
:Poke-i C-64
KLAN SPECTRUM C.D. ... 24
:AY & gry
:Dekoder transmisji radiowej
KLAN AMSTRAD 25
:Operacje dyskowe w systemie CP/M+ (5)
KLAN IBM 29
:MS-Windows
:Stuku puku
:Gmeranie między klawiszami
:Klawisze jeszcze raz
:Wciąż nie DOS-yć o DOS-ie
:Byte
:NARC — program dearchiwizujący
WSZYSTKO DLA WSZYSTKICH 35
GIEŁDA, IBD 38
DROGI BAJTKU 39
DIASTEMOS 40

Do końca roku do szkół trafi 3200 komputerów klasy IBM PC

BELFER Z DYSKIETKĄ

Od dawna w „Bajtku" zajmowaliśmy się stanem i perspektywami tzw. komputeryzacji szkół. Liczne programy w których jesteśmy mocni — gwarantowały dobre samopoczucie ich autorom i smutną prozę codzienności nauczycielom i uczniom, zmagającym się z przedmiotem „informatyka".

Dzieliliśmy się z Czytelnikami wątpliwościami na temat Juniora, pisaliśmy o braku oprogramowania edukacyjnego. Narzekaliśmy na tempo wprowadzanych zmian.

Pod koniec 1990 r. nie jesteśmy optymistami. Wiemy, że poprawa stanu rzeczy nie jest ani łatwa, ani możliwa jednym „Wielkim Skokiem". To co uznajemy za iskierkę nadziei, związane jest z dostrzegalnym wzrostem aktywności rodziców i grup nauczycieli chcących „zrobić coś dla siebie".

Na szczęście Ministerstwo Edukacji Narodowej zrezygnowało z popierania Juniora decydując się na komputery klasy IBM PC. Do końca roku do szkół trafi trzy tysiące dwieście tych komputerów.

Naszym zdaniem największy postęp związany jest z oprogramowaniem edukacyjnym. Coraz więcej firm i placówek naukowych zabrało się za produkcję rodzimego software'u.

Na zorganizowanej w dniach od 17 do 20 września w Błażejewku koło Kórnika VI konferencji **„Informatyka w Szkole"** katalog oprogramowania dydaktycznego doręczony uczestnikom zawierał kilkadziesiąt pozycji, z których kilkanaście można uznać za więcej niż interesujące. Z przyjemnością stwierdziliśmy, że wiele z prezentowanych na łamach „Bajtka" pomysłów — świadomie, lub nie — zostało przez innych wcielonych w życie.

Dotyczy to między innymi idei unifikacji oprogramowania pracującego pod kontrolą systemu operacyjnego CP/M. W roku ubiegłym ukazał się na ten temat w „Bajtku" artykuł pióra Jonasza Mayera. Dziś firma Vulkan z Wrocławia oferuje pakiet oprogramowania użytkowego na Juniora i IBM zgodny z założeniami przedstawionymi przez nas. Mimo, że CP/M przeszedł do historii w krajach zachodnich, to w Polsce z pewnością znajdzie jeszcze wielu amatorów, szczególnie wśród użytkowników Spectrum ze stacją dysków FDD-3000 i Juniora, z którym pożegnano się, ale nie do końca.

Dobrze, że dziś nauczyciele informatyki, chemii i fizyki mają do dyspozycji choć kilkanaście różnych programów edukacyjnych. Nie jest to wiele, ale początek został zrobiony.

Szkoda, że brak jest pomysłów na programy mogące służyć w nauce języka polskiego, historii, geografii czy języków obcych. Z pewnością pojawi się zapotrzebowanie na oprogramowanie mogące wesprzeć nauczycieli ekonomii.

Jedną z największych przeszkód w rozwoju produkcji software'u jest brak ochrony prawnej oprogramowania. Sejm RP nie uchwalił jeszcze nowej ustawy o ochronie prawa autorskiego, która zagwarantowałaby opłacalność produkcji także dla oświaty. Jest to przecież wspaniały klient. Co roku nowi uczniowie wchodzą do klas i odkrywają nieznane obszary wiedzy, także dzięki komputerom.

Za dziesięć lat wszyscy będziemy żyli w XXI wieku. O naszej pozycji w świecie zadecyduje potencjał intelektualny pokolenia dzisiejszych dwunasto-trzynastolatków.

Co „Bajtek" może uczynić, aby pomóc zainteresowanym tą tematyką? Zwiększenie objętości daje nam możliwość publikacji dodatkowych materiałów. Postaramy się w najbliższych numerach zamieścić informacje specjalnie dla nauczycieli, nie zapominając oczywiście o uczniach. W tym celu nawiązaliśmy kontakty z wydawanym przez Ogólnopolską Fundację Edukacji Komputerowej miesięcznikiem „Komputer w Szkole", który jest „Bajtkiem" dla belfrów.

Na łamach naszego pisma zaprezentujemy niektóre tematy poruszane przez kolegów. Będą to opisy programów edukacyjnych i ich recenzje, a także krótkie porady dotyczące wykorzystania komputera w celach innych niż gry. Przedstawimy też nauczycieli, których wielka pasja zaowocowała bardzo interesującymi dokonaniami w dziedzinie oprogramowania.

Ci ludzie stanowią o szansach i przyszłych możliwościach pokolenia, które w pierwszych latach XXI wieku przeżyje swój rozkwit.

Dobrze się stało, że Ministerstwo odeszło od „ręcznego" sterowania rozdziałem środków na informatykę dając większą swobodę lokalnym organom.

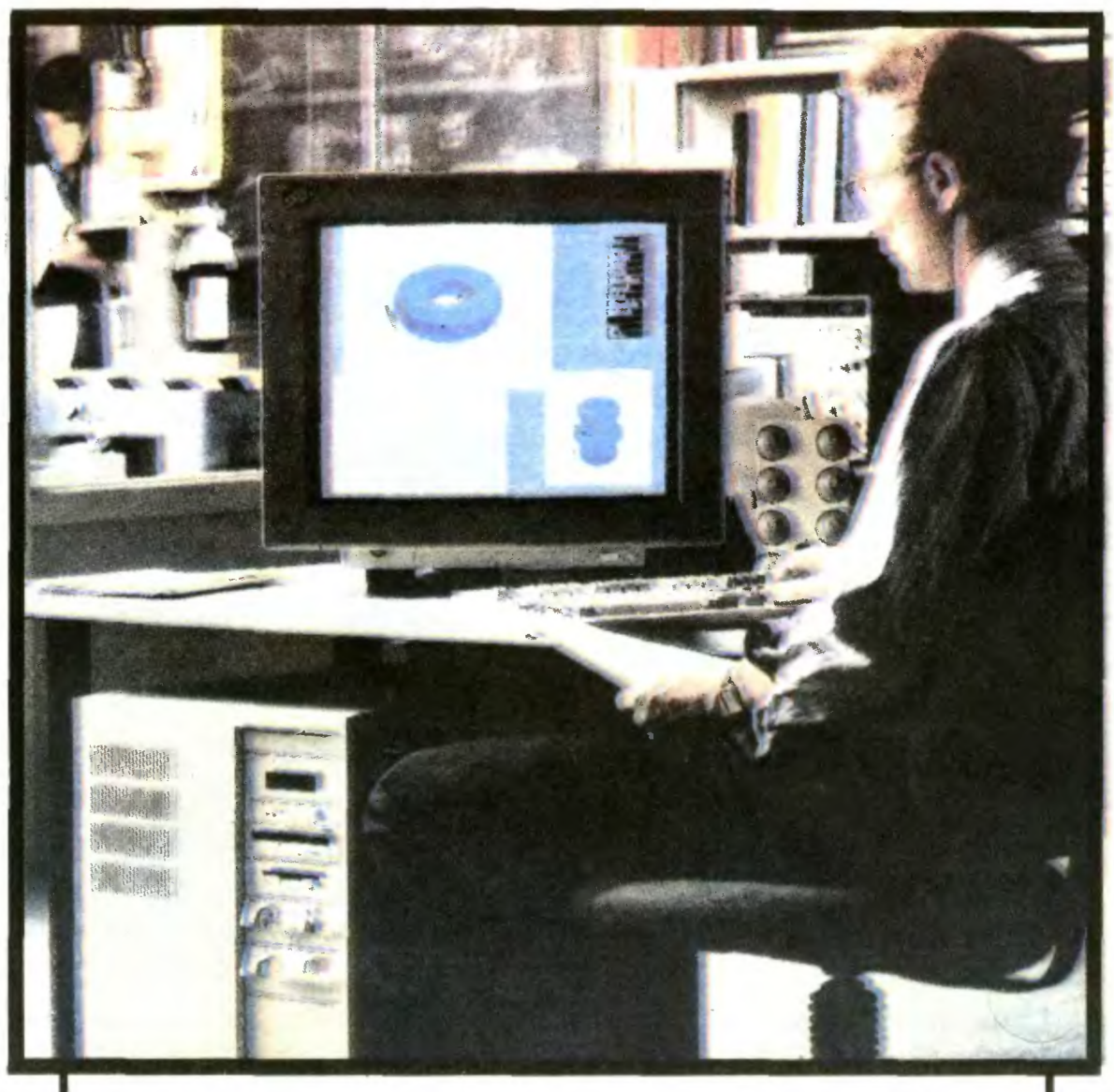
Nie wszystko jednak wygląda optymistycznie. Z coraz większym niepokojem obserwuję zdziczenie obyczajów panujące wśród młodych ludzi, zajmujących się handlem oprogramowaniem. W licznych minigazetkach i informatorach giełdowych pojawiają się teksty gloryfikujące wręcz piractwo programowe. To zły znak. Część odpowiedzialności za ten stan spada na ludzi dorosłych. Brak dobrego prawa jeszcze długo będzie się mścił.

Pozostaje mieć tylko nadzieję, że w drodze do Europy przyswoimy sobie wartości takie jak „własność" i „nie kradnij".

Marek Czarkowski

Redakcja „Bajtka" zwraca się z prośbą do wszystkich zainteresowanych nauczycieli o nadsyłanie swoich opinii dotyczących oprogramowania edukacyjnego z którego korzystacie Państwo w codziennej pracy. Prosimy także o informacje o własnych programach w celu ich rozpropagowania. Chcemy pomóc Państwu w wymianie informacji. Postaramy się odpowiedzieć na wszystkie listy.

MICRO



STACJA ROBOCZA IBM

Rodzina stacji roboczych produkowanych przez koncern IBM powiększyła się o model RISC System/6000 POWERstation 320. Jest to komputer o ogromnych możliwościach.

Szybkość przetwarzania danych sięga 27.5 miliona instrukcji na sekundę (MIPS). Jego architektura oparta została o najnowszy mikroprocesor zaprojektowany przez IBM i wykonany w technologii CMOS.

POWERstation 320 posiada także swe bardziej rozwinięte wersje oznaczone symbolami 520 i 530. Te systemy mają możliwość przetwarzania danych z szybkością 34.5 MIPS.

Omawiane komputery znajdą zastosowanie w instytutach i przemyśle ze szczególnym uwzględnieniem prac projektowych. IBM liczy także na zainteresowanie wyższych uczelni i środowisk naukowych. Cena POWERstation 320 sięga 10000 dolarów.

(mc)

HANDYSCANNER DO C-64

Na rynku niemieckim pojawił się ręczny scanner przystosowany do współpracy z Commodore 64. Można dzięki niemu przenosić na pamięci dyskowe obrazy z rozdzielczością 300 dpi.

Wiadomo, że Commodore 64 jest jednym z najlepiej oprogramowanych komputerów świata. Tym bardziej cieszy pojawienie się oprogramowania scannera. Martwi jego wysoka cena — 498 DM za samo urządzenie.

Scanner jest droższy od komputera — nawet w Polsce, gdzie, jak sądzę, „Baltona" sprzedaje najdroższe C64 na świecie — 207 dolarów za sztukę! A wszystko za sprawą fiskusa, który obłożył komputery podatkami. Pochwalamy zdecydowane działania odpowiednich urzędów państwowych słusznie broniących rodzimych producentów przed zalewem tandetnych produktów „Made in USA, Made in Taiwan" itp. Na scanner proponujemy nałożenie jeszcze większego cła, bo przecież wiadomo, że nasi konstruktorzy pracują nad podobnym urządzeniem dla Juniora już od 45 lat.

(mc)

PROGRAMISTA — SZANTAŻYSTA

Pan dr Ryszard P. z Warszawy, za pośrednictwem redakcji dwóch brytyjskich czasopism komputerowych — PC Plus i Personal Computer World, złożył firmie Clockwork Software propozycję „nie do odrzucenia"

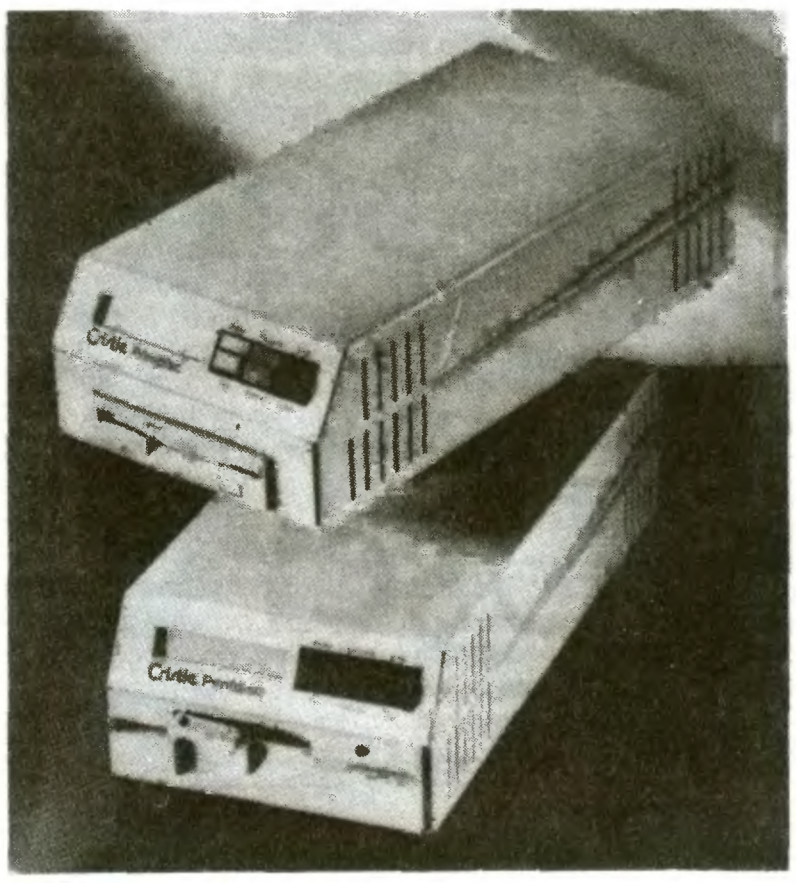
Do lipcowego zeszytu magazynu PC Plus dołączona była dyskietka z programem, w wersji demonstracyjnej, jednego z produktów wspomnianej firmy software'owej. Pan dr P. poświęcił dwa tygodnie na usunięcie zabezpieczeń i w efekcie otrzymał pełną, nie okrojoną wersję komercyjnego programu. Zastosowana metoda okazała się na tyle ogólna, że może być również użyta do innych produktów rozpowszechnianych za pośrednictwem pisma PC Plus.

Zawsze wiedzieliśmy, że mamy doskonałych programistów, nie ustępujących swoim zachodnim kolegom, ale niestety ich moralność często pozostawia wiele do życzenia. Tak też było i w tym przypadku. Pan dr P. nie ograniczył się tylko do głębokiej rozrywki intelektualnej, ale usiłował zdyskontować swój wysiłek w inny, mniej etyczny sposób.

W liście skierowanym do redakcji PC Plus i PCW domaga się on od firmy Clockwork Software dwutygodniowego wynagrodzenia programisty, który zabezpieczał wspomniany program. W przypadku nie spełnienia swoich żądań, grozi rozpowszechnieniem artykułu o roboczym tytule „Clockwork Software products for free", zawierającym praktyczne informacje o łamaniu zabezpieczeń. Kolejną groźbą jest propozycja rozpowszechnienia w sieci FidoNET programu KILL-NIGEL, który w prosty sposób zamienia wersje demonstracyjne w pełne komercyjne programy, sprzedawane przez firmę.

Takie postępowanie nie przynosi nam wszystkim chluby w oczach zachodniej opinii publicznej i zostało określone jako próba szantażu. Nie dość, że nie mamy pieniędzy i w związku z tym nie warto z nami handlować, bo i tak dużo nie kupimy, to nawet jeśli coś nabędziemy, to nie szanujemy praw własności.

(jm)



PRINT DISC

drukuj bez komputera

Większość posiadaczy komputerów w naszym kraju nie dysponuje własną drukarką. Idealnym rozwiązaniem dla nich byłyby punkty usługowe, podobne do kserograficznych, w których dzięki urządzeniu PrintDisc mogliby wydrukować swoje pliki przyniesione na dyskietkach. Produkt ten oferowany przez brytyjską firmę Cristie Electronics Ltd. przypomina dodatkową stację dysków przyłączaną niekiedy do komputerów typu laptop. Praktycznie jest to jednak samodzielny komputer wyposażony oprócz standardowego napędu (5.25" lub 3.5" — zależnie od wersji) także w panel sterujący i złącze szeregowe typu V. 24/RS-232C. Interface typu Centronics jest opcją. PrintDisc umożliwia odczyt plików z komputera IBM PC i ich drukowanie lub rysowanie. Dzięki złączu szeregowemu możliwe jest również przesyłanie do dalszej obróbki także plików z innych komputerów, wyposażonych w to złącze.

W zamiarze projektantów było, aby urządzenie to odciążało stanowiska projektowe typu CAD od czasochłonnych wydruków. Niestety obawiam się, że w naszych warunkach, na przeszkodzie szerszemu rozpowszechnieniu tego gadgetu stoi dość wysoka cena — 900 funtów.

(jm)

MAGAZYN

PRYWATNY monitor

Ten prywatny monitor wyposażony jest w ekran o przekątnej jednego cala, ale ze względu na odległość od gałki ocznej patrzący ma złudzenie obserwowania 12 calowego monitora.

Firma Reflection Technology z Massachussetts dokonała kolejnego kroku na drodze miniaturyzacji. Ekran wykonany w technologii LED pozwala na uzyskanie rozdzielczości trybu CGA. Zasilany jest z baterii o mocy 0,5 wata.

Nie wiemy, jaki jest wpływ tego monitora na wzrok właściciela, ale obawiamy się, że cena 495 dolarów na razie ograniczy liczbę posiadaczy tego urządzenia.

Czy za kilka lat nie spotkamy na ulicy np. fana gier komputerowych noszącego swój komputer jak walkmana na pasku?

(mc)

ATARI TT

Firma Atari nie chce być gorsza od Commodore. W czwartym kwartale tego roku wprowadzi na rynek swój nowy komputer—Atari TT — będący kontynuacją znanej linii ST.

Atari TT jest komputerem działającym w oparciu o 32-bitowy mikroprocesor Motorola 68030 taktowany zegarem 16 MHz. Nowy system operacyjny TOS 030 jest kompatybilny z systemem stosowanym w rodzinie ST — ST/TOS. Jak twierdzi producent, wspaniałe możliwości muzyczne zapewni Yamaha Soundchip YM — 2149. Do tego 6 MB RAM na płycie głównej z możliwością rozszerzenia do 16 MB i kolorowy monitor VGA 14 cali lub monitor monochromatyczny wysokiej rozdzielczości.

To co jest nowością w tej konstrukcji, to możliwość zainstalowania systemu operacyjnego UNIX nazwanego przez producenta „ATX" zgodnego z opracowanym na Uniwersytecie w Berkeley systemem UNIX 5.3.1. Wydaje się, że Atari próbuje konkurować z firmą Commodore ostro lansującą swój nowy produkt—komputer Amiga 3000. Na ile udana dla Tramiela będzie ta konfrontacja, czas pokaże. Dziś wiemy jedynie tyle, że sporo się mówi o Amidze, zaś o Atari TT cicho. *(mc)*

mysz do LOTUSA

Power Mouse firmy ProHance Technology, Inc. stanowi skrzyżowanie programowalnego kalkulatora i myszy. 40 klawiszy tego urządzenia pozwala na zaprogramowanie 240 funkcji istotnie ułatwiających korzystanie z programów kalkulacyjnych.

Ta nietypowa mysz została zaprojektowana przede wszystkim do współpracy z arkuszem Lotus 1-2-3. Do komputera typu IBM PC podłączona jest przez złącze RS-232C, a obsługę programową zapewnia specjalizowany program rezydentny, niezgodny ze standardem firmy MicroSoft.

Szerszemu wykorzystaniu tego dość użytecznego urządzenia, o dużych potencjalnych możliwościach, przeszkadza aktualny brak właściwego oprogramowania sterującego (ang. drivers) do większości programów oferowanych przez rynek.

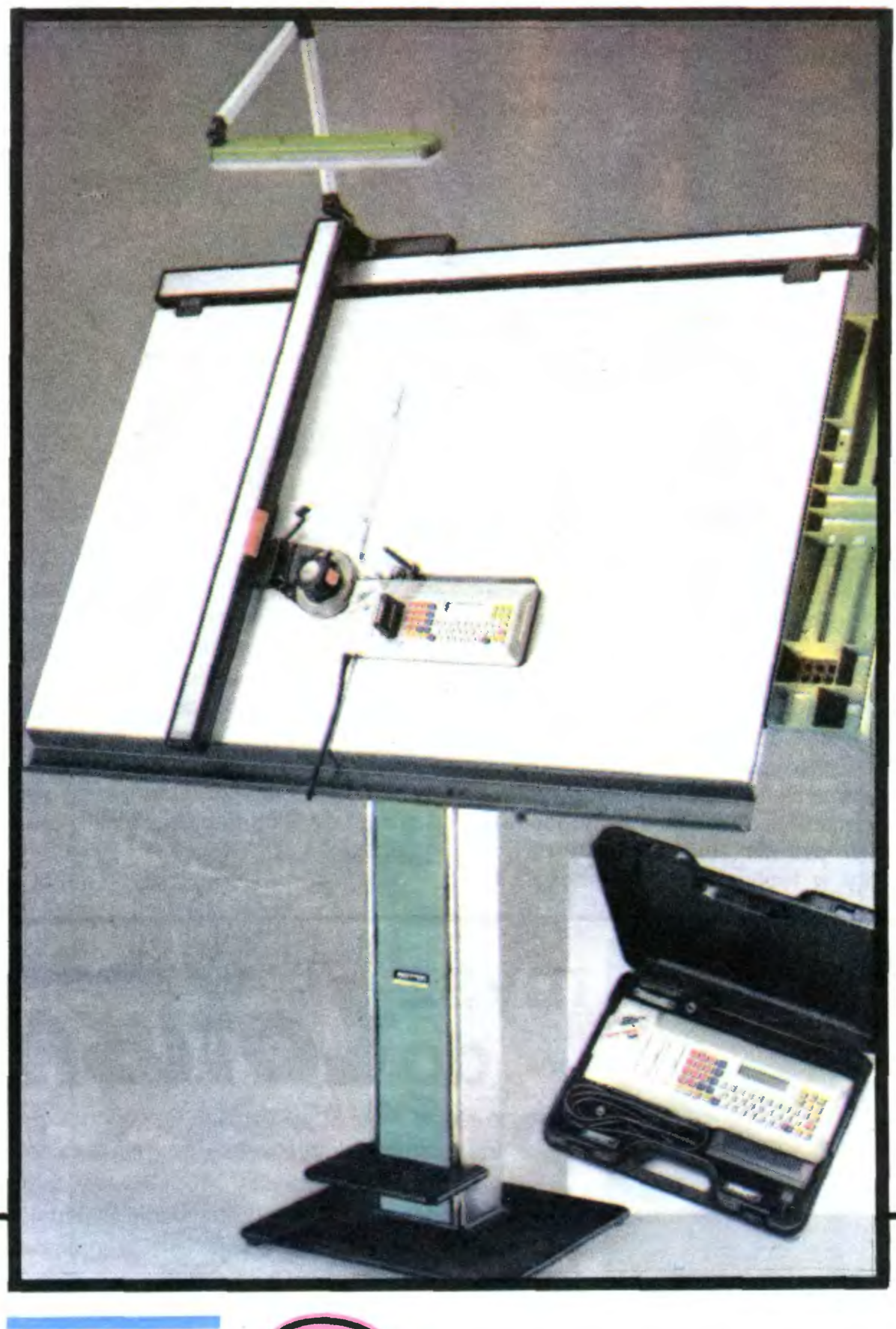
(jm)

FOTOPLOTER FIRMY GLASER

Fotoploter Galaxy 4000 szwajcarskiej firmy Glaser jest urządzeniem, które umożliwia precyzyjne wykonywanie skomplikowanych płytek drukowanych. Optyczna głowica wyposażona w diody typ LED pozwala na kreślenie obrazów z precyzją ±0.005 mm i rozdzielczością 0.02 lub 0.01 mm na obszarze 500 x 690 mm. Do naświetlania używane są światłoczułe folie lub specjalne płyty szklane o dowolnej grubości.

Do sterowania fotoplotera wymagany jest dość rozbudowany i szybki komputer klasy IBM PC/AT 386. Wymiana informacji odbywa się przez złącze szeregowe RS 232C w językach HPGL (Hewlett-Packard Graphics Language) i PostScript. Sądzę, że nie bez znaczenia są też możliwości poligraficzne takiego fotoplotera.

(jm)



mini PLOTER DLA KREŚLARZA

Niedocenioną rewelacją warszawskiej wystawy Komputer-90 był też mini ploter włoskiej firmy Neolt S.p.A. Urządzenie to będąc skrzyżowaniem programowalnego kalkulatora i mini plotera jest doskonałym narzędziem dla osób zajmujących się kreśleniem rysunków technicznych. Niewielkie rozmiary (415*180*45 mm.) i niewielki ciężar (1.8 kg) oraz możliwość przymocowania miniplotera do przykładnicy stołu kreślarskiego pozwalają na poprawianie i opisywanie wykonanych wcześniej dużych rysunków.

Powierzchnia rysowania wynosi 60*250 mm, a mocowany rapidograf, poruszający się z szybkością od 5 do 40 mm/s kreśli dowolne figury geometryczne i litery o wielu skalowanych rozmiarach i pochyleniach. Mimo, że urządzenie jest całkowicie autonomiczne i programowalne przez użytkownika, może ono również współpracować z dowolnym komputerem wyposażonym w złącze szeregowe.

Klawiatura zawierająca 58 klawiszy, wyświetlacz ciekłokrystaliczny na 40 znaków, 8KB pamięci RAM i wymienne pakiety z programami zapewniają komfortowe warunki pracy. Również sama idea tego nowatorskiego urządzenia jest głęboko humanistyczna. Przyzwyczailiśmy się do kalkulatorów i łatwiej jest nam je zaakceptować niż komputery, których możliwości i działanie wydają się nam czasem bardzo skomplikowane. W przypadku opisywanego mini plotera nie mamy takich obciążeń i sądzę, że urządzenie to znajdzie swoją drogę do pracowni projektowych także i w naszym kraju.

(jm)

Carry — I

Jedną z niewielu rzeczy, która wzbudziła nasz zachwyt podczas tegorocznej wystawy Komputer-90 był komputer klasy IBM PC/AT tajwańskiej firmy Flytech Technology Co., Ltd. Carry-i, bo taka jest jego nazwa, to niewielkie pudełko o rozmiarach 240*185*45 i wadze 2—3 kg. Jego sercem jest procesor 80286 taktowany zegarem 12 MHz i pozwalający na pracę z pamięcią RAM o rozmiarze 1Mb bez taktów oczekiwania (zero wait state).

Standardowym wyposażeniem jest napęd 3.5" 1.44 Mb, dwa złącza szeregowe i jedno równoległe. Dodatkowo można dodać drugą, taką samą stację dysków lub dysk twardy 20 MB. Zasilacz znajduje się w oddzielnej obudowie, a sam komputer sprzedawany jest także z małym monitorem o średnicy ekranu 9" i typową klawiaturą o zmniejszonych rozmiarach. Udostępnione dwa standardy karty graficznej — Hercules I CGA — umożliwiają zastosowania zarówno profesjonalne, jak i rozrywkowe.

Estetyczna obudowa, naprawdę ciesząca oczy, niewielki ciężar i poręczność skłaniają do przypuszczenia, że będzie to chętnie kupowany przez ludzi sprzęt o parametrach laptopa, a bez pretensji do jego ceny.

(jm)

MYSZ — PIÓRO —

MYSZ – pióro to nowość na rynku amerykańskim. Urządzenie jest w pełni kompatybilne z Microsoft Mouse i być może zyska uznanie w oczach właścicieli komputerów.

Zasada działania myszy-pióra jest identyczna jak zwykłej myszy. Nowość chętnie współpracuje z takimi programami jak Lotus 1-2-3, dBASE III+ i WordPerfect. Jedynie cena zmusza do zastanowienia — 129 dolarów!

Producent — International Machine Control Systems Inc. 1332 Vendels Cir. Paso Robles, California 93446.

(mc)

KOLOROWY SCANNER

Tylko 995 dolarów kosztuje kolorowy scanner firmy Sharp przystosowany do współpracy z komputerami Macintosh II. Wymagane są 4MB RAM na płycie głównej komputera i kolorowy monitor. JX—100, bo takie jest oznaczenie fabryczne scannera, jest mniejszy od znanego już JX—450 i, jak sądzi producent, zainteresuje właścicieli Mac-ów parających się na niewielką skalę Desktop Publishing.

Właścicieli komputerów klasy IBM PC i Amiga zainteresuje wiadomość, że firma Sharp pracuje obecnie nad przystosowaniem swego scannera do współpracy z tymi komputerami.

(mc)

DRUKOWANIE W SZPALTACH

Standardowe edytory pozwalają zwykle na wydruk tekstu w jednej szpalcie. Większe możliwości dają programy typu DTP, służące m.in. do łamania tekstu na kilka szpalt. Umiejętność taka bywa przydatna także przy drukowaniu listingów, zwłaszcza jeśli zależy nam na oszczędności papieru.

Przedstawiony obok program w Turbo Pascalu pozwala na wydruk dowolnego zbioru tekstowego w zadanej ilości szpalt. Sam program główny nie jest skomplikowany i zawiera wywołanie trzech procedur: **InitDefault**, **GetFileNames** i **Process**. Pierwsza z nich inicjalizuje wartości domyślne wydruku: rozmiar strony 126 kolumn, 82 wiersze i dwie szpalty. Druga z procedur pobiera nazwę pliku przetwarzanego i dalsze parametry — nazwę pliku wyjściowego i ewentualne rozmiary strony oraz podział na szpalty. Ostatnia z procedur realizuje właściwe przetwarzanie tekstu poprzez wypełnianie macierzy L odpowiadającej drukowanej stronie.

Podstawowa procedura programu korzysta z trzech pomocniczych procedur: **PrintPage**, **NextSpalte** i **NextLine**. Pierwsza z nich drukuje wypełnioną tablicę L, a dwie następne sterują przejściem odpowiednio od nowej szpalty lub linii.

Jeśli program skompilujemy nadając mu nazwę NP, to składnia jego wywołania jest następująca:

NP fn1 [fn2 C L S]

gdzie **fn1** — nazwa pliku przetwarzanego,
fn2 — nazwa pliku wyjściowego (podanie LST: — kieruje wydruk na drukarkę),
C — liczba kolumn na stronie minus 1,
L — liczba wierszy na stronie minus 1,
S — liczba szpalt, na jakie dzielimy tekst (minus 1).

Parametry podane w nawiasie nie są obowiązkowe i nie podanie ich powoduje przyjęcie wartości domyślnych.

Jeśli zależy nam na maksymalnym zagęszczeniu wydruku, to przed uruchomieniem programu należy wysłać na drukarkę sekwencję: **#27#48#27#15** lub dodać na początku programu (po otwarciu pliku **g**) instrukcję **Write (g,...)** zawierającą ten ciąg bajtów.

Dla drukarek 10" zadany rozmiar strony — stałe MaxCol i MaxLin w programie — jest wystarczający. Dla drukarek 15" lub dłuższej strony stałe te należy zwiększyć.

Jonasz Mayer

```
Program PrintFileInSpaltes;
{************************************************}
{           (C) JM 24 Jan 1990                   }
{ Drukuje plik tekstowy w kilku szpaltach.       }
{ Skladnia:                                      }
{           NP fn1 [fn2 C L S]                   }
{************************************************}
const
     MaxCol = 125;   { Max liczba kolumn/strone }
     MaxLin =  99;   { Max liczba wierszy/strone}
type
     AnyStr= String [128];
     Lines = array [0..MaxCol] of char;
     Tab   = array [0..MaxLin] of Lines;
var
     CurMaxCol,  { Akt. max. liczba kol/str    }
     CurMaxLine, { Akt. max. liczba wiersz/str }
     CurMaxSpa,  { Akt. max. liczba szpalt/str }
     CL,         { liczba kolumn/szpalte }
     Line,       { Akt. nr. linii   }
     Spalte,     { Akt. nr. szpalty }
     Col,        { Akt. nr. kolumny }
     Page        { Akt. nr. strony  }
                 : Integer;
     L           : Tab;
     f,g         : text;

procedure pError (s : AnyStr; flag : boolean);
{************************************************}
{************************************************}
begin
  Write ('Error. ',s);
  if flag
  then begin
         write('. Program halted.'); Halt;
       end;
end; { of pError }

procedure InitDefault;
{************************************************}
{************************************************}
begin
  CurMaxCol  := 125;     { Columns - 1 }
  CurMaxLine :=  81;     { Lines - 1   }
  CurMaxSpa  :=   1;     { Spaltes - 1 }
  CL := (CurMaxCol + 1) div (CurMaxSpa + 1) - 2;
end; { of InitDefault }

Procedure Par (Var X : integer; n : integer);
{************************************************}
{************************************************}
var S : string [3];
    z, code : integer;
begin
  s := ParamStr (n);
  if Length (S)>0
  then begin
         VAL (S,z,code);
         if code=0 then x := z;
       end;
end; { of PAR }

procedure GetFileNames;
{************************************************}
{************************************************}
var ch : char;
    Fname, Gname : String [20];
begin
  if ParamCount=0 then pError ('No params Specified', True);
  if ParamCount=1
  then Gname := 'Lst:'          { printer as default }
  else Gname := ParamStr (2);
  Fname := ParamStr (1);
  Par (CurMaxCol,3);
  Par (CurMaxLine,4);
  Par (CurMaxSpa,5);
  CL := (CurMaxCol + 1) div (CurMaxSpa + 1) - 2;
  Assign (f, Fname);     {$I-}
  Reset (f);             {$I+}
  If IOresult<>0 then pError ('Input file not found',True);
  Assign (g,Gname);      {$I-}
  reset  (g);            {$I+}
  if IOresult=0
  then begin
         pError ('Output file already exists.'+
                 ' Overwrite (Y/N) ?',false);
         repeat
           read (kbd,ch);       ch := UpCase (ch);
         until ch in ['Y','N'];
         if ch='N' then pError ('',True);
       end;
  rewrite (g);                  writeln;
end; { of GetFileNames }

procedure PrintPage (Var Page,Spalte,Line : integer);
{************************************************}
{************************************************}
var i,j,
    MaxLines, MaxCols : integer;
begin
  If Spalte=0
  then MaxLines := Line - 1
  else MaxLines := CurMaxLine;
  if MaxLines>0
  then begin
         Page := Page + 1;
         Writeln ('Page: ',Page);
       end;
  For i := 0 to MaxLines
  do begin
       if i<Line
       then MaxCols := Spalte * (CL+2) + CL
       else MaxCols := (Spalte-1) * (CL+2) + CL;
       For j := 0 to MaxCols
       do write (g, L[i,j]);    writeln (g);
     end;
  Line := 0;  Spalte := 0;  Col := 0;   write (g,#12)
end; { of PrintPage }

procedure NextSpalte (var Spalte, Line : integer);
{************************************************}
{************************************************}
begin
  Line := 0;
  Spalte := Spalte + 1;
  if Spalte = CurMaxSpa+1
  then PrintPage (Page, Spalte, Line);
end; { of NextSpalte }

procedure NextLine (var Spalte,Line,Col : integer);
{************************************************}
{************************************************}
var i : Integer;
begin
  for i := Col to CL + 1
    do L [ Line, Spalte*(CL+2) + i] := ' ';
  Col := 0;
  Line := Line + 1;
  if Line=CurMaxLine+1 then NextSpalte (Spalte,Line);
end; { of NextLine }

procedure Process;
{************************************************}
{************************************************}
var ch : char;
    i  : integer;
begin
  Line   := 0;  Spalte := 0;
  Col    := 0;  Page   := 0;
  while not eof (f)
  do begin
       while not eoln (f)
       do begin
            read (f,ch);
            L [Line,Spalte * (CL+2) + col] := ch;
            col := col + 1;
            if (col = CL+1) and (not eoln(f))
            then NextLine (Spalte,Line,Col);
          end; { of while not eoln (f) }
       readln (f);
       NextLine (Spalte,Line,Col);
     end; { of while not eof (f) }
  PrintPage (Page,Spalte,Line);
  Close (f);
  Close (g);
end; { of Process }

{************************************************}

begin { main }
  InitDefault;
  GetFileNames;
  Process;
end.

{************************************************}
```

AMIGA 3000

Pod hasłem „Nowa Amiga 3000: moc obliczeniowa Macintosha II za cenę Commodore" firma Commodore Business Machines Inc. wprowadziła na rynek swe najmłodsze dziecko — komputer Amiga 3000.

O tym komputerze mówiło się od dawna. Jak wspomina Guy Kewney w „Personal Computer World", już dwa lata temu dyrektor techniczny Commodore Henry Rubin zaprezentował mu Amigę 3000 w tajnym pokoju laboratorium firmy w Atlancie. Rubin zapewniał dziennikarza "PCW", że ten komputer nie będzie gorszy od stacji roboczych firmy Sun.

Czy te nadzieje się spełniły?

W prasie fachowej w maju i czerwcu znalazły się omówienia walorów technicznych Amigi. Podkreślano śmiałość rozwiązań technicznych — „Byte" numer majowy i kłopoty z oprogramowaniem — „Personal Computer World" numer czerwcowy. Trudno nie odnieść wrażenia, że Amigę 3000 potraktowano z rezerwą, spowodowaną być może czekaniem na wyraźny sygnał od użytkowników. „Jeśli zaczną kupować — to znaczy, że jest dobrze".

Sama Amiga jest dziś z całą pewnością komputerem domowym lat osiemdziesiątych, a być może nawet dziewięćdziesiątych. Szalony sen Jaya Minera — technicznego geniusza zakochanego w szybkiej grafice i efektach — spełnił się.

Kiedy w roku 1984 po raz pierwszy publicznie przedstawiono komputer Amiga 1000, było to cacko dla fachowców, oczarowanych możliwościami graficznymi dzieła Minera. Po sześciu latach doczekaliśmy się całej rodziny Amig, której zwieńczeniem ma być ta z numerem 3000.

Przyjrzyjmy się jej bliżej.

Jest to komputer produkowany w dwóch wersjach — pierwsza taktowana jest zegarem o częstotliwości 16 MHz, a druga — zegarem 25 MHz.

Obie wersje zostały zbudowane na bazie mikroprocesora Motorola 68030. Jest to mikroprocesor trzydziestodwubitowy. W ten sposób firma Commodore przekroczyła kolejną ważną dla producentów sprzętu barierę. Oznacza to także wyraźne zwiększenie mocy obliczeniowej komputera. Przypomnijmy, że Amiga 2000 taktowana jest zegarem o częstotliwości 8 MHz, a jej mikroprocesor — to Motorola 68000. Wrażenie robi także pamięć RAM — 2 MB z możliwością rozszerzenia praktycznie do 128 MB, a teoretycznie nawet do 2000! Tyle pamięci może „obsłużyć" mikroprocesor Motorola.

Producent zaopatrzył komputer w kolorowy monitor 14 cali o wspaniałej rozdzielczości, istnieje możliwość zastosowania monitora monochromatycznego.

Jedyne, co powstrzymuje potencjalnych klientów od natychmiastowego sięgnięcia po książeczki czekowe i karty kredytowe, to cena, kształtująca się obecnie na poziomie 3495 dolarów za wersję z zegarem 16 MHz i 3995 dolarów za wersję z zegarem 25 MHz.

Wydaje się, że cena komputera jest zbyt wysoka. Tym bardziej, że krytycy zwracają uwagę na słaby punkt — oprogramowanie.

Przed oficjalną premierą Amigi 3000 zapowiadano, że nowy komputer firmy Commodore będzie pracował pod kontrolą systemu operacyjnego UNIX. Tymczasem okazało się, że jedynie przewidziano możliwość zainstalowania UNIX-a, wyposażając pierwsze wersje w system operacyjny Amiga DOS 2.0, będący kontynuacją Amiga DOS 1.3.

Jak na ambicje dorównania stacjom roboczym, może to być za mało. Jak pisze Guy Kewney w podsumowaniu swojego artykułu w „Personal Computer World", może się okazać, że dla użytkowników komputerów pracujących pod kontrolą UNIX-a cena nie będzie miała decydującej roli, a o wiele ważniejsze okażą się stare nawyki i przyzwyczajenia sprawiające, że niełatwo rozstaną się oni ze znanymi dobrze komputerami firm Sun czy DEC.

Myślę, że jest to opinia sceptyka. Bob Ryan — autor *cover story* na temat Amigi 3000, zamieszczonego w "Byte", podkreśla kierunek rozwoju komputera — multimedia. To, jego zdaniem, przyszłość, którą widzi firma Commodore.

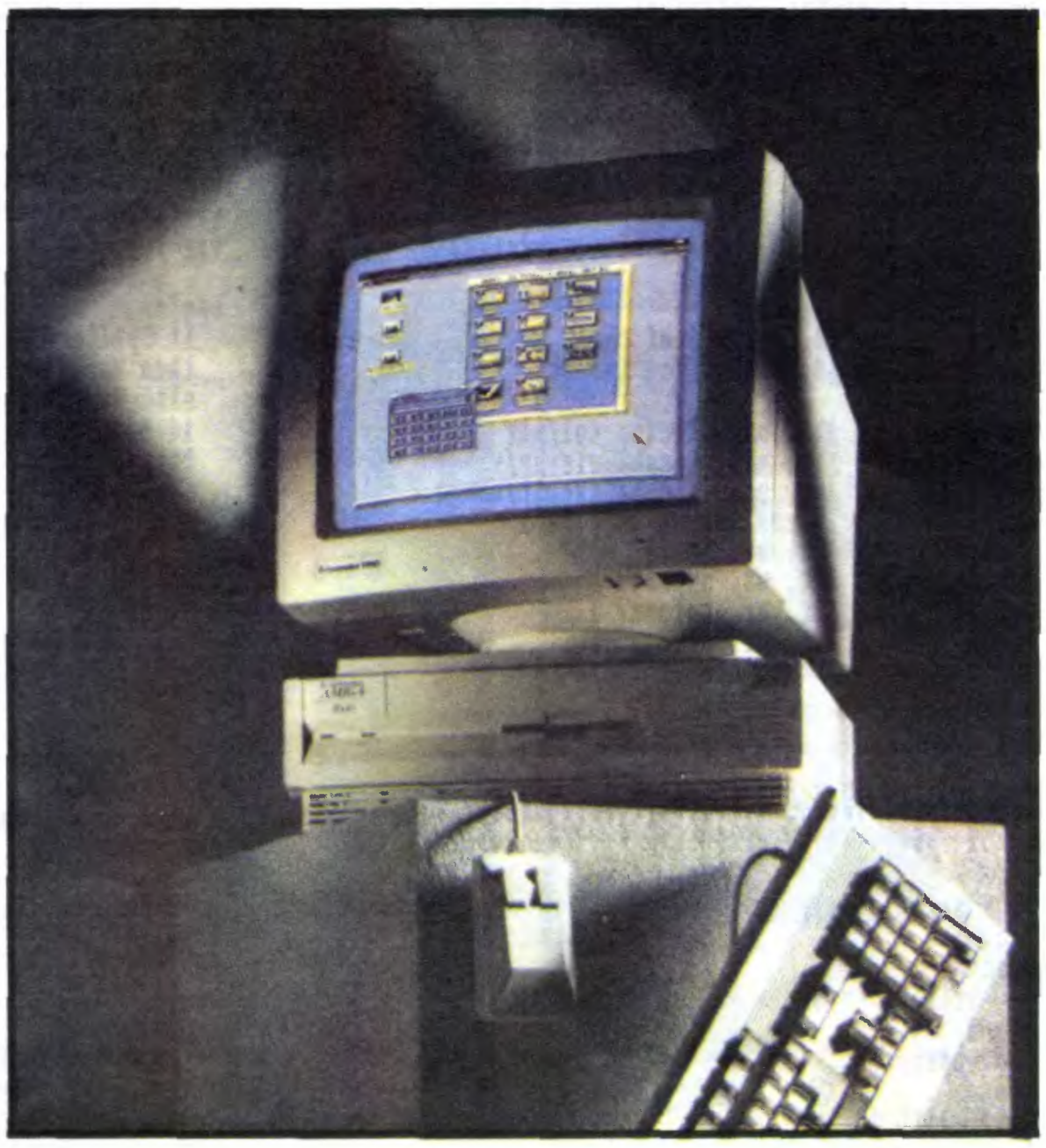
Nie sposób dziś rozstrzygnąć, która z opinii okaże się trafna. Amiga sama w sobie stała się legendą lat osiemdziesiątych i z pewnością fakt ten nie jest bez znaczenia dla perspektyw handlowych nowego produktu.

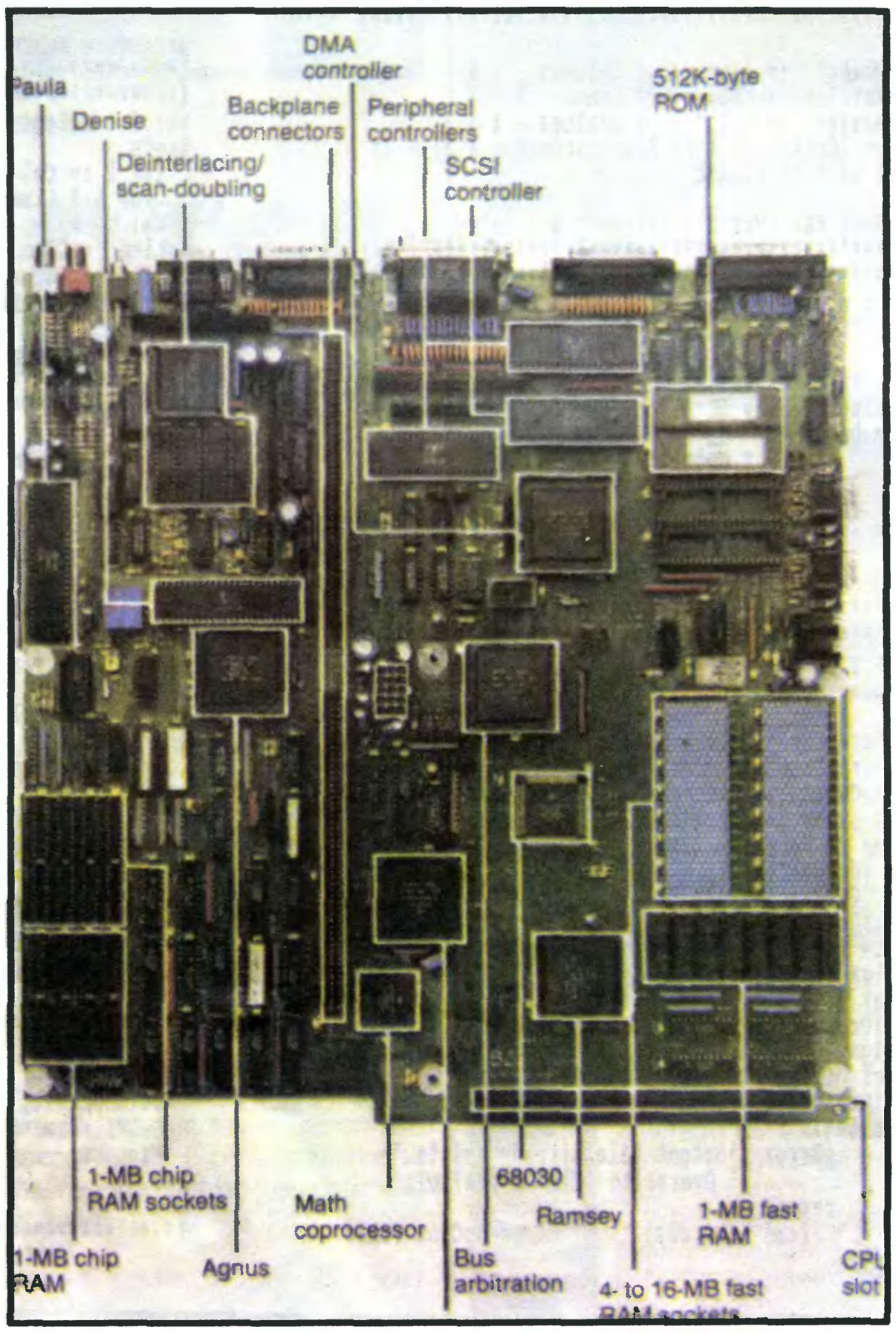
Bardzo wielu obecnych właścicieli Amigi 500 czy 2000 zechce kontynuować swój związek z rodziną kupując, o ile stan konta na to zezwoli, Amigę 3000. Boję się jednak, że zbyt często widzimy w tym komputerze jedynie supermaszynkę do gier. Co się stanie, jeśli pewnego dnia na rynku pojawią się „gierki" na Amigę 3000 wykorzystujące wspaniałe bez wątpienia możliwości techniczne tego komputera? Czy aby nie zostanie przeniesiony na wyższy poziom wyścig między firmami produkującymi gry?

Wiem, że w Polsce Amiga 500 zdobywa coraz więcej zwolenników. Szkoda, że jej cena w sklepach Baltony jest przeraźliwie wysoka — ponad 700 dolarów. Mam nadzieję, że ten krótki z konieczności opis usatysfakcjonuje fanów Amigi.

Więcej informacji znajdą oni pod adresem
Commodore Business Machines, Inc.
1600 Wilson Dr.
West Chester, PA 19380
(215) 431-9100

Marek Czarkowski





MAŁY POMOCNIK

Programując w Commodore Basic możemy komunikować się z dyskiem. Zwykle odbywa się to poprzez kanały otwierane podaniem nazwy pliku dyskowego do otwarcia.

Zdarza się jednak, że z wielu plików należy wybrać tylko jeden, o właściwym typie i z nazwą oraz pozbawiony tzw. jokerów. One głównie przeszkadzają w operacjach, gdyż DOS nie akceptuje obarczonych nimi nazw.

Mój programik, napisany na C128, pomoże łatwo otwierać różnorakie pliki. Przy odrobinie chęci można uruchomić go i na C64, po niewielkich przeróbkach, jak np. zamknięcie plików (1.100):fori=1 topeek(152):closepeek(600+1):nexti:end

Program nazwałem roboczo procedurą 900903. Na listingu jest ona uwikłana w nieco większą strukturę programową, która jest w moich programach generalnie niezmienna. Polega ona na tym, że linie 10—100 zawierają start programu — początek i koniec, 110—200 obróbkę błędów, dalej umieszczone są podprogramy, a od linii 1000 zaczyna się główny program. Pozwala to np. na łatwe zamknięcie wszystkich otwartych plików — goto100.

Podprogram 900903 wywoływany jest po wyszukiwaniu pliku i potwierdzeniu operatora, że nazwa jest poprawna. Analizuje on ostatni katalog w buforze stacji i jeżeli wykazany był tylko jeden plik, to nazwa przyporządkowana jest zmiennej x$, a typ zmiennej y$. Jeśli zaś plików było więcej lub żaden, to zmienne te są puste. Można więc użyć podprogramu podając od razu nazwę i typ w x$ i y$.

Na wejściu podprogramu potrzebne są jeszcze: u% — numer stacji, d% — numer napędu, f% — wolny numer pliku i c% — kanału wyszukiwane są automatycznie.

Procedura jest szybka — (ok. 3 sek) i daje się ładnie kompilować dzięki użyciu liczb integer.

Piotr Niemiec

```
 filename`900903

10 pn$="`filename`900903":rem by piotr niemiec
20 poke53280,0:poke53281,0:print"■■■":z$="y"
30 open1,0:goto1000
40 :
50 :
60 :
70 :
80 :
90 :
100 fori=1topeek(152):closepeek(600+i-(peek(48)<8)*265):nexti:end
110 rem error-correction (only on c128!)
120 :
130 :
140 :
150 :
160 :
170 :
180 :
190 :
200 :
210 :
220 datadel,seq,prg,usr,rel,cbm,all:g%=len(x$):forg=1tog%:g$=""
230 ifmid$(x$,g,1)="="theng$=mid$(x$,g+1,1):x$=left$(x$,g-1):g=g%
240 nextg:restore:forg=0to6:ready$:ifleft$(y$,1)<>g$thennextg
250 f%=0:g%=(peek(48)<8)*265:h%=0:ifpeek(152)=0goto300
260 forf=1topeek(152):ifpeek(610+f-g%)<>iu%orpeek(620+f-g%)<>96goto290
270 print"■■ sorry, the input-channel on your disk- ";:print"■■ unit ";
280 printright$(str$(iu%+1e2),2)", drive"id%"■ is already occupied ! ■":stop
290 f%=f%+(f%<peek(600+f-g%))*(f%-peek(600+f-g%)):nextf
300 f%=f%+1:h$=chr$(0)
310 openf%,iu%,0,"$"+mid$(str$(id%),2)+":"+x$+",p,r"
320 forg=0to31:get#f%,g$:nextg:f=1:f$="":rem ■header-loop
330 get#f%,g$,g$,a$,b$:x$=mid$(str$(asc(a$+h$)+256*asc(b$+h$)),2)+" "
340 forg=4to30:get#f%,g$:x$=x$+chr$(asc(g$+h$)):nextg:get#f%,g$:g$=mid$(x$,25,3)
350 if(y$=g$)or(y$="all")or(right$(x$,2)=h$+h$)thenf=f+1:f$=f$+x$
360 iff<3andrig∧t$(x$,2)<>h$+h$goto330
370 closef%:f=-17*(f=3andright$(f$,2)=h$+h$):x$=mid$(f$,7,f):ifx$=""goto390
380 y$=right$(x$,1):x$=left$(x$,f-1):ify$<>chr$(34)thenf=f-1:goto380
390 y$=mid$(f$,25,-3*(x$>"")):return
400 :
410 :
420 :
1000 print"■■input unit:■ 8  ■■■■";:input#1,u$:print:iu%=val(u$):ifiu%<0goto100
1010 ifiu%<8oriu%>11goto1000
1020 print"■"tab(27)"■drive:■ 0  ■■■■";:input#1,d$:print:id%=val(d$)
1030 ifid%<0orid%>1goto1020
1040 print"■■filename:■              ■■■■■■■■■■■■";:input#1,x$:print:ifx$=""goto100
1050 iflen(x$)>16thenprint"■■■■":goto1040
1060 print:gosub220:ifx$=""thenprint"■■■ no valid filename! ■":goto1080
1070 print"■filename:■ "x$tab(27)"■filetype:■ "y$;:ifpeek(48)<8thenprint
1080 input"■■ok (y/n)■  n■■■■";y$:ify$=z$goto100
1090 goto1000
1100 end
ready.
```

KLUB C 128

Jeden z ostatnich numerów miesięcznika "64er" poinformował o klubie specjalizującym się głównie w Commodorach 128. Jest to według tego miesięcznika największy klub tego rodzaju w Europie. Klub ten stosuje tzw. taryfę ulgową dla „wschodniaków" — półroczna składka wynosi dla nich tylko 12.50 DM i jest to zaledwie koszt wydania ich kwartalnika, który otrzymuje każdy członek klubu. Poza tym, wszyscy wstępujący do klubu otrzymują bezpłatnie dyskietkę z wieloma ciekawymi programami, a także, gdy zapłacą 10 DM, mogą dostać dyskietkę ekstra z jeszcze ciekawszą ofertą. Klub dysponuje sporą ilością programów typu "public domain", które z czystym sumieniem można nabyć. Niestety, dużym utrudnieniem jest dostarczenie nagranych dyskietek z RFN-u do Polski. Jak wiadomo, jedynym pewnym sposobem na to jest znalezienie sobie „wujka" w RFN, który będzie miał okazję je przywieźć. Oto adres klubu:

Club 128er Aktuell
Christian Evers
Fritz-Reuter-Strasse 31
Oer-Erkenschwick
D-4353

TYLKO DLA DYSKO MANÓW

Jeśli dopiero od niedawna masz Commodore 64 i nie zdobyłeś dotąd żadnego dobrego programu zarządzającego stacją dysków, to wpisz ten program. Nie jest to hit na miarę naszej ery, ale może okazać się pomocny w wielu prozaicznych sytuacjach.

Obsługa programu jest prosta. Jeśli jest już wpisany i nagrany na dowolnym dysku, wgraj go do pamięci. Następnie włóż do stacji dyskietkę, którą chcesz sprawdzić, i napisz RUN.

Komputer załaduje do pamięci informacje odczytane z dysku i zacznie je wyświetlać w podanej niżej kolejności. Muszę w tym miejscu zaznaczyć, że używanie angielskich określeń zostało wymuszone przez nieprzetłumaczalność niektórych angielskich zwrotów, nie mających w języku polskim dostatecznie dobrych odpowiedników.

Na początku ukaże się napis **LOADING HEADER BLOCK** — ładowanie danych. Po chwili zobaczysz tabelę programów znajdujących się na dysku — **FILE**, wraz z informacjami o nich. Są to: **TYPE** (typ pliku), **SIZE** (długość w blokach), **TRACK** i **SECTOR** (numer ścieżki i sektora, na których znajduje się początek programu).

Gdy zostaną wyświetlone wszystkie tytuły, komputer zaczeka na wciśnięcie dowolnego klawisza i przejdzie do następnej części — **VITAL STATISTICS** (w wolnym tłumaczeniu — statystyka zapisania dysku). Są to informacje dotyczące liczby programów na dysku — **NUMBER OF PROGRAMS,** ilości zapisanych bloków — **BLOCKS USED,** liczby nie zapisanych bloków — **BLOCKS FREE** oraz liczby wolnych bloków na osiemnastej ścieżce — **DIRECTORY BLOCKS FREE**.

Po wyświetleniu tych informacji i wciśnięciu przez użytkownika dowolnego klawisza, komputer przejdzie do ostatniej części programu, noszącej tę samą nazwę co poprzednia.

Na ekranie ukażą się numery poszczególnych ścieżek, a pod nimi ilość wolnych jeszcze na niej bloków. Tak jak zapewne wszystkim wiadomo, Commodore formatuje dyskietkę w sposób następujący: na pierwszej i następnych szesnastu mamy po 21 bloków wolnych, na osiemnastej tylko 17 bloków, a potem od 19 do 24 — 19 bloków, od 25 do 30 — 18 bloków i wreszcie od 31 do 35 — 17 bloków. Jeśli dana ścieżka ma nie zmienioną ilość wolnych bloków, np. ścieżka nr 1 będzie miała ich 21, to liczba 21 będzie napisana na biało. Jeśli zaś warunek ten nie będzie spełniony, kolor będzie ciemniejszy.

Ciekawostką jest to, że program wykrywa też pliki zapisane dawniej i skasowane, na których miejsce nie zostało zapisane nic nowego. Ponieważ plik taki nie posiada pierwszej litery swojej nazwy, więc komputer wydrukuje tylko jego dane liczbowe — długość i miejsce zapisu na dysku.

I na koniec jeszcze kilka rad o wpisywaniu programu. Barwy niezbędne do jego poprawnego działania napisane są w nawiasach, podobnie zresztą jak i inne specyficzne funkcje. Tak więc napis (**RED**) oznacza kombinację CONTROL + 3, (**WHT**) to po prostu CONTROL + 2. Trochę inne znaczenie mają liczby napisane w nawiasach ze znaczkiem dolara. ($1) wpisujemy wciskając CONTROL + A, ($3) CONTROL + C. Funkcję (**DEL**) wklepujemy przy użyciu następującej kombinacji: otwieramy cudzysłów, wciskamy kilka razy INST, a potem DEL dotąd, aż pozostanie tylko jedna literka T w inwersie. (**REV**) i (**OFF**) to odpowiednio CONTROL + 9 i CONTROL + 0. (**DOWN**) to inaczej CURSOR DOWN. Napis (**SPC**) jest skrótem słowa SPACE i wraz z liczbą charakteryzują długość odstępu, jaki należy zostawić.

Łukasz Czekajewski

```
100 POKE53280,0:POKE53281,0:GOTO440
110 GET#8,A$:IFA$=""THENA$=CHR$(0)
120 RETURN
130 INPUT#15,E,E$,T,S
140 IFE>0THENPRINT"($3)(RED)"E;E$" "T","S:END
150 RETURN
160 GOSUB110
170 DN$=DN$+A$
180 RETURN
190 GET#8,A$:IFA$=""THEN190
200 IFSTTHEN430
210 IFASC(A$)<128THENF$="(DEL)":GOTO260
220 IFASC(A$)AND56THEN190
230 F$="  ":F=(ASC(A$)AND63):F=F+2*(F-1):IFASC(A$
)AND64THENP$="(RVS)<(OFF)"
240 F$=MID$("SEQPRGUSRREL",F,3)
250 GOSUB110
260 T$=RIGHT$("  "+STR$(ASC(A$)),2)
270 GOSUB110
280 S$=RIGHT$("  "+STR$(ASC(A$)),2)
290 N$=""
300 FORA=1TO16
310 GET#8,A$:N$=N$+A$
320 NEXT
330 FORA=1TO9:GET#8,A$:NEXT
340 GOSUB110
350 BL=ASC(A$)
360 GOSUB110
370 BL=BL+ASC(A$)*256:IFF$<>"(DEL)"THENBT=BT+BL
380 B$=RIGHT$("   "+STR$(BL),3)
390 PRINT"(WHT)";:IFF$="(DEL)"THENPRINT"($3)(BLK
)";
400 POKE212,128
410 PRINTN$;:POKE212,0
420 PRINT"($3)(YEL)"P$"  "F$"    "B$"    "I$"     "S
$
430 RETURN
440 PRINT"(CLR)(YEL)(RVS)LOADING HEADER BLOCK"
450 OPEN15,8,15,"I0":GOSUB130
460 BT=0:DIMBM$(35)
470 OPEN8,8,1,"$,S,R":GOSUB130
480 FORA=1TO2:GET#8,A$:NEXT:A=0
490 FORA=1TO35
500 GOSUB110
510 BM=BM+ASC(A$):BM$(A)=RIGHT$("  "+STR$(ASC(A$)
),3)
520 GET#8,A$:GET#8,A$:GET#8,A$
530 NEXT
540 DN$="(CLR)(RVS)(YEL)DISK:  (RVS)"+CHR$(34)
550 FORA=1TO16:GOSUB160:NEXT
560 DN$=DN$+CHR$(34)
570 FORA=1TO15:GOSUB160:NEXT
580 PRINTBN$:B=0
590 PRINT"(DOWN)(WHT)FILE NAME(SPC9)($3)(GRN)TYP
E($3)(RED)  SIZE($3)(BLU)  TRACK SECTOR(DOWN)"
600 GOSUB190:B=B+1:IFF$<>"(DEL)"THENPT=PT+1
610 IFSTTHEN630
620 IFB<20THEN600
630 POKE198,0:PRINT"(RED)(RVS)(DOWN)(DOWN)PRESS
ANY KEY(OFF)";
640 GETK$:IFK$=""THEN640
650 IFST=0THEN580
660 GOSUB130:CLOSE8:CLOSE15
670 PRINT"(CLR)(WHT)(SPC10)VITAL STATISTICS"
680 PRINT"(SPC11)"
690 PRINT"(YEL)NUMBER OF PROGRAMS ="PT
700 PRINT"BLOCKS USED ="BT
710 BF=BM-VAL(BM$(18))
720 PRINT"BLOCKS FREE ="BF;
730 A$=""
740 IF664-BT<>BFTHENA$="($3)(RED)BLOCK COUNT MIS
MATCH(YEL)"
750 PRINTA$
760 PRINT"DIRECTORY BLOCKS FREE ="BM$(18)
770 PRINT"(RVS)(RED)(DOWN)(DOWN)PRESS ANY KEY(OF
F)":WAIT198,1
780 A$="(GRN)-(SPC36)-"
790 PRINT"(DOWN)(GRN)($3)($1)*30($3)(HOME)"
800 PRINT"(DOWN)(GRN)-FREE TRACK SECTORS:(SPC17)
-"
810 PRINT"(GRN)-(SPC36)-"
820 PRINT"(GRN)-($3)(YEL)TRACK:  1  2  3  4  5
6  7  8  9 10(GRN)-"
830 PRINT"(GRN)-($3)(BLU)TOTAL:";:FORA=1TO10:PRI
NT"($3)(BLU)";:IFVAL(BM$(A))=21THENPRINT"($3)(WH
T)";
840 PRINTBM$(A);:NEXT:PRINT"(GRN)-":PRINTA$
850 PRINT"(GRN)-($3)(YEL)TRACK: 11 12 13 14 15 1
6 17 18 19 20(GRN)-"
860 PRINT"(GRN)-($3)(BLU)TOTAL:";:FORA=11TO17:PR
INT"($3)(BLU)";:IFVAL(BM$(A))=21THENPRINT"($3)(W
HT)";
870 PRINTBM$(A);:NEXT
880 FORA=18TO20:PRINT"($3)(BLU)";:IFVAL(BM$(A))=
19THENPRINT"($3)(WHT)";
890 PRINTBM$(A);:NEXT:PRINT"(GRN)-":PRINTA$
900 PRINT"(GRN)-($3)(YEL)TRACK: 21 22 23 24 25 2
6 27 28 29 30(GRN)-"
910 PRINT"(GRN)-($3)(BLU)TOTAL:";:FORA=21TO24:PR
INT"($3)(BLU)";:IFVAL(BM$(A))=19THENPRINT"($3)(W
HT)";
920 PRINTBM$(A);:NEXT
930 FORA=25TO30:PRINT"($3)(BLU)";:IFVAL(BM$(A))=
18THENPRINT"($3)(WHT)";
940 PRINTBM$(A);:NEXT:PRINT"(GRN)-":PRINTA$
950 PRINT"(GRN)-($3)(YEL)TRACK: 31 32 33 34 35(S
PC15)(GRN)-"
960 PRINT"(GRN)-($3)(BLU)TOTAL:";:FORA=31TO35:PR
INT"($3)(BLU)";:IFVAL(BM$(A))=17THENPRINT"($3)(W
HT)";
970 PRINTBM$(A);:NEXT:PRINT"(SPC15)(GRN)-"

READY.
```

KILKA PRAKTYCZNYCH RAD DLA KUPUJĄCYCH AMIGĘ

SŁOWO O AMIGACH

Postanowiłem: kupuję Amigę!

Sprawa wydaje się prosta. Algorytm zaczyna się od sprawdzenia ilości pieniędzy. Jeśli jest ich dość, można już wysłać kogoś do krajów bogatszych, lub też udać się na giełdę.

Rozterki rozpoczynają się w momencie uświadomienia sobie, że istnieje kilka typów owego gloryfikowanego komputera. Zazwyczaj informacje kończą się na tym, że KickStart 1.2 jest lepszy, bo wszystkie programy chodzą. Takie ujęcie sprawy nie jest właściwe.

KickStart jest to kość będąca podstawą dla systemu operacyjnego Workbench i często jest z nim utożsamiana. Nowsza wersja, KickStart 1.3 ma to do siebie, że niektóre programy odmawiają posłuszeństwa. Jest za to kilka zalet, o których nie można zapominać. Przede wszystkim KickStart 1.3 może bootować twardy dysk, skuteczniej chroni przed wirusami oraz posiada ułatwienia dla programujących w asemblerze.

Nie tak dawno pojawiła się nowa wersja KickStartu, oznaczona 1.3.2 i gorąco polecam ją wszystkim zainteresowanym. Najprostszy sposób wykrycia to uruchomienie RSI CEBIT DEMO.

Jako, że nie z samego KickStartu Amiga się składa, przejdźmy do płyty głównej. Istnieją dwie wersje — nowa i stara. Nowa ma miejsce na dołożenie 0.5MB bez konieczności kupowania cartridge'a co daje oszczędności. Oprócz tego istniejący na płycie 1MB może funkcjonować jako CHIP-RAM.

Nową płytę można rozpoznać po tym, że przy resecie komputera LED nie gaśnie, a jedynie przygasa.

Na nowej płycie znajduje się również nowy Blitter dający szersze możliwości graficzno-animacyjne. Już w tej chwili dostępne są programy pracujące tylko z nowym Blitterem.

Przy zakupie Amigi radzę również zwrócić uwagę na pracę stacji — zbyt głośna źle o niej świadczy.

Jeśli chodzi zaś o Amigę 2000, to z istniejących na rynku wersji A, B i C stanowczo polecam tę ostatnią. Różnice polegają na przydziale pamięci na CHIP-RAM i FAST-RAM; wersja A jest pod tym względem najgorsza. Wersja B, choć już z 1MB CHIP-RAM-u, może nie posiadać KickStartu 1.3 (ważne dla twardodyskomanów). Polecam wersję C, jako dotychczas najlepiej dopracowaną.

Mateusz Krause

JĘZYK "C" DLA NAJMŁODSZYCH

CZYLI ÓSMY WIECZÓR Z CZARNOKSIĘŻNIKIEM

Podczas poprzedniego spotkania stwierdziliśmy, że podprogram kończąc swoją pracę likwiduje swe zmienne lokalne. Unika w ten sposób kolizji zmiennych z programem wywołującym go. Oczywiście, ponowne wywołanie podprogramu musi znowu spowodować powstanie zmiennych lokalnych. Przysłonią one zmienne lokalne programu wywołującego, pozwalając im w ten sposób zachować ich dotychczasową wartość. Kolizji nie będzie nawet wtedy, gdy podprogram wywoła sam siebie.

Dawno temu, w pewnej szkole nauczyciel matematyki chciał na lekcji napisać list. Postanowił zająć uczniów prostym, ale pracochłonnym zajęciem. Uczniowie mieli dodać do siebie wszystkie liczby naturalne od 1 do 100. Prawie wszyscy uczniowie zaczęli tę pracę w oczywisty sposób: 1 + 2 + 3 + ... + 98 + 99 + 100. Spróbujmy napisać program, który będzie implementacją tego algorytmu, dla dowolnej liczby N.

W tym celu trzeba zauważyć, że suma pierwszego elementu równa jest jemu samemu S1 = 1.

Następne sumy wyglądają tak:

$$S2 = 1 + 2 = S1 + 2$$
$$S3 = 1 + 2 + 3 = S2 + 3$$
$$S4 = 1 + 2 + 3 + 4 = S3 + 4$$

W ogólności jest SN = S(N−1) + N. Jeśli więc chcemy napisać funkcję, która policzy naszą sumę, wystarczy, że będzie ona znała wynik dla jednego elementu (np. dla 1) oraz będzie wiedziała, jak obliczyć wynik dla każdego następnego.

Oto program, który obliczy i napisze taką sume:

```
#include <stdio.h>
#include <predef.h>

majster
{calkowite s;
 powtarzaj (;;)
 {pisz("\n Jaka liczba?");
  czytaj ("%d",&s);
  gdy (s>=0) pisz ("S%d = %d",s,suma(s));
  przeciwnie pisz ("Co Ty?!");
 }
}
suma (licz)
calkowite licz;
{gdy (!licz) oddaj (0);
 przeciwnie oddaj (licz*suma(licz-1));
}
```

Łatwc zauważyć, że podprogram "suma" wywołuje sam siebie, tylko z inną wartością parametru — argumentu. Taki sposób nazywamy rekurencją. W optyce łatwo ją wywołać ustawiając dwa lustra naprzeciw siebie lub kamerę tv przed monitorem, który pokazuje obraz z tej kamery.

Wspominaliśmy, że podprogram wywoływany tworzy własne zmienne lokalne, przysłaniając zmienne lokalne programu wywołującego. Gdy kończy pracę, likwiduje swoje zmienne, odsłaniając zmienne lokalne programu wywołującego z ich dotychczasowymi wartościami. Nie ma obawy o kolizję. Jest jednak obawa o to, że stos zmiennych lokalnych, utworzony na skutek kolejnych rekurencyjnych wywołań, wypełni całą pamięć komputera. Ponownie wywołany podprogram nie będzie już mógł działać. Jest to wypadek, z którym trzeba się liczyć, dlatego jako zmienne lokalne i jako parametry uznajemy te wielkości, które mają istotny wpływ na działanie rekurencji. Wszystkie inne potraktujmy jako zmienne globalne. W tym przypadku ma to sens.

Napiszmy teraz program, który będzie liczył nie sumę, ale iloczyn N liczb naturalnych. 1 * 2 * 3 ... * (N—1) * N. Wielkość taką nazywamy N-silnia i piszemy N!. Jedna mała uwaga: 0! = 1.

Program, który za pomocą rekurencji liczy się silnię dowolnej liczby, korzysta z własności, że:

$$1! = 1$$
$$2! = 2 * 1 = 2 * 1!$$
$$3! = 3 * 2 * 1 = 3 * 2!$$
$$4! = 4 * 3 * 2 * 1 = 4 * 3!$$

Ogólnie $N! = N * (N-1)!$

Oto ten program:

```
#include <stdio.h>
#include <predef.h>

majster
{calkowite s;
 powtarzaj (;;)
 {pisz("\n Jaka liczba?");
  czytaj ("%d",&s);
  gdy (s<0) pisz ("\n Co Ty?!");
  przeciwnie pisz ("\n%d! = %d",s,silnia(s));
 }
}
suma (licz)
calkowite licz;
{gdy (!licz) oddaj (1);
 przeciwnie oddaj (licz*silnia(licz-1));
}
```

Program jest prosty i czytelny. Ma jednak słaby punkt. Jeśli działa na małym komputerze, np. małe Atari, to dla liczb większych niż 7 wypisuje jako wynik liczby, które nie są dobrym wynikiem. Zakres liczb na takim komputerze wynosi od −32768 do 32767. Łatwo zobaczyć, że 7! = 5040, a więc 8! = 40320. Jest to liczba spoza zakresu, nastąpiło więc przeniesienie na bit znaku. Komputer ten nie sygnalizuje błędu i program nie może tego zauważyć. W takiej sytuacji możemy wybrać jedno z dwu wyjść — zwiększyć umiejętności komputera, zmieniając go na większy, albo zwiększyć swoje umiejętności.

My wybierzemy drugą możliwość. Oto sposób, który umożliwi nam obliczenie silni dla większych liczb. Zastosowany algorytm musi pozwalać na łatwe obliczenie poprawnego wyniku i jego wydruk. Oczywiście, trzeba go zapamiętać w tablicy zmiennych całkowitych.

Wcześniej wspomniano, że arytmetyka szesnastobitowa (taka jest w małym Atari) mieści liczby najwyżej do 32767. Dla nas ważne jest, że najwyższą potęgą dziesiątki, jaka się mieści, jest 10000. Zredukujmy sztucznie zakres z 32676 do 9999. Z chwilą jego przekroczenia następuje zmniejszenie wyniku o 10000 i przeniesienie (dodanie) jedynki do następnej zmiennej w tablicy wyników. Zmienna ta, i każda następna, będzie także gromadziła liczby od 0 do 9999.

Zwykłe mnożenie elementów tablicy wyników przez kolejne liczby może dać wyniki na tyle duże, że nadmiary wymkną się nam spod kontroli. Dotyczy to oczywiście dużych liczb, dla takich bowiem chcemy napisać nasz program.

Ostatniemu niebezpieczeństwu także można zaradzić — zastosujemy algorytm mnożenia binarnego. Nazwa może brzmi groźnie dla początkujących programistów, ale nie taki diabeł straszny...

Prześledzimy, jak takie mnożenie wygląda w przypadku dwu liczb, np. 25 * 13. Obie liczby można zapisać w systemie binarnym. Trzeba je tylko rozłożyć na sumę liczb będących kolejnymi potęgami dwójki. Otrzymamy:

$25 = 16 + 8 + 1 = 2^4 + 2^3 + 2^0$ oraz $13 = 8 + 4 + 1 = 2^3 + 2^2 + 2^0$

Na tych pozycjach, które odpowiadają występującym potęgom, wpisujemy jedynkę, na pozostałych zero. Ostatecznie otrzymujemy:

25(d) = 11001(b) oraz 13(d) = 1101(b)

Teraz wykonujemy mnożenie tak, jakby były to liczby dziesiętne:

```
      1 1 0 0 1
    * 1 1 0 1
      1 1 0 0 1
    1 1 0 0 1
+ 1 1 0 0 1
1 0 1 0 0 0 1 0 1
```

Każdy wynik cząstkowy jest faktycznie liczbą, którą mnożymy, przesuniętą w lewo o odpowiednią ilość pozycji (bitów). Gdy na którejś pozycji w liczbie, przez którą mnożymy, wystąpiło zero, to takie przesunięcie jest pomijane. Wyniki te dodajemy jak liczby binarne. Nie jest to trudne — wystarczy pamiętać, że suma dwu jedynek daje zero z przeniesieniem jedynki na następną pozycję. Ostatecznie mamy więc:

$25 * 13(d) = 11001 * 1101(b) = 101000101 = 2^8 + 2^6 + 2^2 + 2^0(d) = 256 + 64 + 4 + 1 = 325$

Literki (d) i (b) upewniają nas, że jesteśmy w systemie dziesiętnym lub binarnym. Mnożąc w taki sposób nie zgubimy żadnego nadmiaru — przeniesienia.

Tablica wyników jest wciąż ta sama dla kolejnych wywołań rekurencyjnych, dlatego warto potraktować ją jako zmienną globalną, a nie parametr wywołania.

Oto program, który całą tę prozę opisuje w języku „C":

```
#include <stdio.h>
#include <predef.h>
#define LIM 8
calkowite s[LIM];
majster
{tekst l[10];
 calkowite i,z,n;
 robgdy (pisz ("\n Jaka liczba?"),czytajwiersz(l)>0
 {n=val(l);
  gdy (n<0) {pisz("\n Co Ty?!");cdn;}
  powtarzaj (i=0;i<LIM;i++)
  s[i]=(i==LIM-1);
  silnia(n);
  pisz("\n%d! = ",n);
  z=0;
  powtarzaj (i=0;i<LIM;i++)
  gdy (z) pisz ("%04d",s[i]);
 }
}
silnia(licz)
calkowite licz;
{calkowite l,j,i;
 calkowite rh,rl,wh,wl;
 gdy (licz<2) oddaj;
 powtarzaj (i=0;i<LIM;i++)
 gdy (s[i])
 {l=licz;rl=s[i];
  wh=wl=rh=0;
  powtarzaj (j=1;l;j<<=1,rh<<=1,rl<<=1)
  {gdy (rl>9999){rl-=10000;rh++;}
   gdy (l&j)
   {wl+=rl;wh+=rh;
    gdy (wl>9999){wl-=10000;wh++}
    l^=j;
   }
  }
  s[i]=wl;
  gdy(wh)
   gdy(i)
   {s[i-1]+=wh;
    j=1;
    robgdy(s[i-j]>9999)
    gdy (i-j>0){s[i-j]-=10000;s[i-j-1]++;j++;}
    przeciwnie {pisz("\n Nadmiar");oddaj;}
   }
   przeciwnie {pisz("\n Nadmiar");oddaj;}
  }
 silnia(licz-1);
}
```

Mały komentarz należy się instrukcji wydruku wyników. Otóż tylko "najstarszy", niezerowy element tablicy wyników pisze się z pominięciem początkowych zer. Wartości wszystkich następnych zmiennych zawierają liczby czterocyfrowe. Muszą więc one być napisane na czterech pozycjach z uwzględnieniem wszystkich zer, także początkowych. Standard języka „C" przewiduje taką sytuację. Należy przed specyfikacją formatu napisać zero. W naszym przypadku jest to formuła %04d. Mówi ona, że parametr procedury należy napisać jako czterocyfrową całkowitą liczbę dziesiętną z ewentualnym uzupełnieniem zerami, jeśli liczba ma mniej niż cztery cyfry. Zera te oraz ewentualne spacje pojawiają się przed liczbą, jest więc ona dosunięta do prawej krawędzi pola wydruku. Jeśli chcemy dosunąć liczbę do lewej strony, to spacje muszą pojawić się za liczbą. W tym przypadku trzeba przed specyfikacją formatu napisać minus.

Niech zmienna z = 17. Mogą zaistnieć następujące sytuacje:

pisz("=%d=",z);	wydrukuje	=17=
pisz("=%5d=",z);	,,	=___17=
pisz("=%05d=",z);	,,	=00017=
pisz("=%-5d=",z);	,,	=17___=

Niestety, kompilator Deep Blue C napisany na małe Atari nie jest pełną implementacją standardu i w tym przypadku zamiast zer wstawi spacje. Są dwa wyjścia. Pierwsze to poprawić bibliotekę podprogramów kompilatora. Tak właśnie postąpił autor tego cyklu i jego funkcja "pisz" reaguje na specyfikację formatu zgodnie ze standardem. Druga możliwość to dopisywać zera "na piechotę".

Na domiar złego, we wspomnianym kompilatorze nie zaimplementowano funkcji „czytaj — scanf". Z tego powodu wczytywanie danych odbywa się dzięki wywołaniu kilku funkcji. Jest to bardzo niewygodne, wymaga bowiem od programisty każdorazowo organizowania procesu wczytywania.

Oto program, który zadziała poprawnie na kompilatorze "DBC"

```
#include <predef.h>
#define LIM 8
calkowite s[LIM];
majster
$(tekst l[10];
  calkowite i,z,n;
  robgdy (pisz("\nJaka liczba?");czytajwiersz(l)>0
  $(n=val(l);
    gdy(n<0) $(pisz("\nCo Ty?!");cdn;$)
    powtarzaj (i=0;i<LIM;i++)
     s[i]=(i==LIM-1);
    silnia(n);
    pisz ("\n%d!= ",n);
    z=0;
    powtarzaj (i=0;i<LIM;i++)
    $(gdy (z&&s[i]<1000)
        $(pisz("0");
          gdy (s[i]<100)
             $(pisz("0");
               gdy(s[i]<10) pisz("0");
             $)
        $)
      gdy (z!=s[i]) pisz("%d",s[i]);
    $)
  $)
  silnia(licz)
  calkowite(licz);
  $(calkowite l,j,i;
    calkowite rh,rl,wh,wl;
    gdy (licz<2) oddaj;
    powtarzaj (i=0;i<LIM;i++)
     gdy (s[i])
     $(l=licz;rl=s[i];
       wh=wl=rh=0;
       powtarzaj (j=1;l;j<<=1,rh<<=1,rl<<=1)
       $(gdy(rl>9999)
           $(rl-=10000;rh++;$)
         gdy (l&j)
            $(wl+=rl;wh+=rh;
              gdy(wl>9999)
                $(wl-=10000;wh++;$)
              l^=j;
            $)
       $)
       s[i]=wl;
       gdy(wh)
        gdy(i)
          $(s[i-1]+=wh;j=1;
            robgdy(s[i-j]>9999)
               gdy(i-j>0)
                 $(s[i-j]-=10000;s[i-j-1]++;j++;$)
               przeciwnie
                 $(pisz("\nNadmiar");oddaj;$)
          $)
        przeciwnie $(pisz("\nNadmiar");oddaj;$)
    $)
    silnia(licz-1);
$)
```

Jak widać, o tym, czy dany język programowania jest wygodny czy nie, decyduje także jakość biblioteki podprogramów, w jaki jest wyposażony kompilator tego języka.

Historia z nauczycielem ma swój ciąg dalszy. Nauczyciel był bardzo zdziwiony, gdy po kilku sekundach jeden z uczniów podał dobry wynik 5050. Metoda, dzięki której uzyskał on tak szybko wynik, była prosta. Zamiast dodawać liczby po kolei, jak leci, poustawiał je w pary: pierwszą z ostatnią (tzn. 100), drugą z drugą od końca (tzn. 99), trzecią z trzecią od końca (tzn. 98), itd. itd. pięćdziesiąta z pięćdziesiątą od końca (tzn. 51). Par takich jest pięćdziesiąt, suma każdej pary wynosi 101. Wynik jest iloczynem 50 * 101. Takie mnożenie można wykonać w pamięci. Ogólny wzór na sumę N elementów ciągu arytmetycznego (a takim są liczby naturalne), w którym różnica między kolejnymi elementami jest stała (w naszym przypadku 1) wygląda tak:

$$Sn = \frac{(a1 + aN) * N}{2} = \frac{(1 + 100) * 100}{2} = 101 * \frac{100}{2}$$

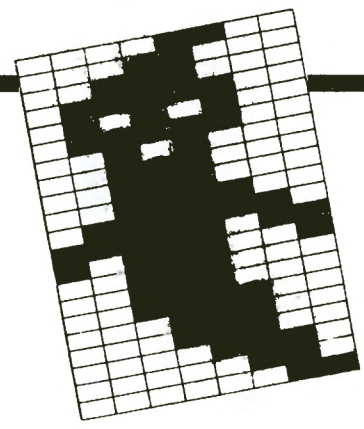
W ten sposób uczeń ten odkrył bardzo ważną własność ciągu arytmetycznego. Mając kilkanaście lat, chłopiec ten został mianowany profesorem, a w matematyce „narozrabiał" jeszcze trochę.

Opowiedziałem wam tę historię, żeby na prostym przykładzie pokazać działanie rekurencji. Chciałem także sprowokować was do myślenia. Czasami warto chwilę pomyśleć, żeby później mniej się narobić (i ewentualnie zostać profesorem).

Programowanie komputerów jest moją pasją, ale nade wszystko jestem ZWOLENNIKIEM MYŚLENIA.

Mieczysław Płacheta

MINI — EDYTOR DUSZKÓW



Projektowanie obiektów grafiki graczy i pocisków, czyli duszków, jest zajęciem dosyć kłopotliwym, jeśli używa się kartki papieru. Lecz komputer, który będzie z tych duszków później korzystał, może znacznie ułatwić pracę.

Zamieszczony program pozwala na tworzenie duszka w polu o wymiarach 8 na 21 punktów. Poruszanie kursora po polu jest realizowane naciskaniem klawiszy ze strzałkami (bez ‹CONTROL›). Klawisz ‹RETURN› powoduje zapalenie lub zgaszenie wybranego punktu. Gotowy wzór przenosi się do obszaru pamięci duszka przez naciśnięcie ‹ESC› — ukazuje się on wtedy w górnej części ekranu. Natomiast jeśli efekt naszej pracy jest niezadowalający, pozostaje wcisnąć ‹SHIFT›+‹CLEAR› i można duszka tworzyć od początku.

Program jest skonstruowany tak, aby nawet początkujący programista mógł poszerzyć jego możliwości. Duszek jest zapamiętywany w zmiennej A$, której zawartość można później zapisać w celu wykorzystania w innym programie. Można także po prostu nacisnąć ‹RESET›, wpisać wiersz

F.X=1 TOLEN (A$):?ASC (A$ (X,X); ",";:N.I:?0

i wyświetlone na ekranie liczby przepisać do instrukcji DATA, a to już potrafimy wykorzystać (patrz seria artykułów o duszkach w „Bajtku").

Michał Widera

```
QM    0 REM Edytor duszkow - wersja 1.0
CC    1 REM Michal Widera, Adam Hetmanski
HZ    2 REM (c) 1990, Bajtek
NI    3 REM
LU    100 DIM A$(22),SPRITE(8,22),DANE(22),B
      IN$(8):CLOSE #1:OPEN #1,4,0,"K:"
IY    110 POKE 106,PEEK(106)-8
EC    120 POKE 82,0:GRAPHICS 0:POKE 53248,0
JH    130 PB=PEEK(106)
ZS    140 FOR A=1 TO 22:FOR B=1 TO 8
PX    150 SPRITE(B,A)=0:NEXT B:NEXT A
WP    160 FOR I=PB*256+1024 TO PB*256+2048:P
      OKE I,0:NEXT I
SC    170 FOR A=1 TO 22:A$(A,A)=CHR$(255):NE
      XT A
IV    180 FOR A=0 TO 7:BIN$(8-A,8-A)=CHR$(2^
      A):NEXT A
IA    190 POKE 54279,PB:POKE 53277,2
IA    200 POKE 53248,90:POKE 559,58
RW    210 GOSUB 400
AB    220 ? "┌──────────┐":FOR X=1 TO 22:? "|
                 |":NEXT X:? "└──────────┘";:X=1:Y
      =1
DJ    230 IF K=42 THEN X=X+1:IF X>8 THEN X=1
FT    240 IF K=43 THEN X=X-1:IF X<1 THEN X=8
YU    250 IF K=45 THEN Y=Y-1:IF Y<1 THEN Y=2
      2
CP    260 IF K=61 THEN Y=Y+1:IF Y>22 THEN Y=
      1
ZF    270 POSITION X,Y:? CHR$(159);
KH    280 Z=PEEK(93):GET #1,K
JE    290 IF K=27 THEN GOSUB 340
YH    300 IF K=125 THEN RUN
FL    310 IF K=155 THEN IF Z=0 THEN POSITION
       X,Y:? "*";:SPRITE(X,Y)=1
WB    320 IF K=155 THEN IF Z<>0 THEN POSITIO
      N X,Y:? " ";:SPRITE(X,Y)=0
NF    330 GOTO 230
HQ    340 FOR A=1 TO 22:SUMA=0
NK    350 FOR B=1 TO 8
GP    360 SUMA=SUMA+SPRITE(B,A)*ASC(BIN$(B,B
      ))
XB    370 NEXT B:DANE(A)=SUMA
YW    380 A$(A,A)=CHR$(DANE(A))
DJ    390 NEXT A
FR    400 FOR A=1 TO LEN(A$):POKE PB*256+105
      9+A,PEEK(ADR(A$)+A-1):NEXT A
ZC    410 RETURN
```

TOTOLOTEK

Zasady gry w toto-lotka znają niemal wszyscy. Wygranie większej kwoty jest jednak dosyć trudne. Zamiast więc wydawać pieniądze w kolekturze zagraj w nasz program. Nic wprawdzie nie możesz wygrać, ale również nic nie stracisz, a satysfakcja taka sama.

Program napisany jest w Turbo Basic'u XL. Należy przepisać go uważnie (korzystając z „Edytora Basic'a") i zapisać na kasetę lub dyskietkę. Teraz już można uruchomić program i rzucić się w szpony hazardu. Obsługa programu jest prosta i nie wymaga dodatkowego komentarza.

Powodzenia!

Leszek Stróżowski

```
NV   1 REM TOTO-LOTEK
CV   2 REM LESZEK STROZOWSKI
XO   3 REM COPYRIGHT (C) BAJTEK
NJ   4 REM
CU   10 GRAPHICS 0:POKE 566,158:POKE 731,1
PD   20 POKE 752,1:EXEC POL:POKE 756,156
PQ   70 EXEC PLANSZA:DIM K(8),T(8),P$(18):P
     $="                  "
YC   75 EXEC TYT
BR   80 FOR R=1 TO 7:K(R)=0:T(R)=0:NEXT R:I
     T=0
OX   90 EXEC KUPON
ZV   95 COLOR 2
EN   100 FOR P=1 TO IS
VZ   110 IF P<7 THEN TEXT 20,183,P:TEXT 28,
     183," LICZBA ?"
OY   115 IF P=7 AND IS=7 THEN TEXT 20,183,"
     DODATKOWA?"
WN   120 TEXT 120,183,"  ":GOSUB 240
QS   130 IF KEY=155 OR KEY=126 OR KEY=48 TH
     EN 120
VN   135 SOUND 0,50,10,10:PAUSE 3:SOUND
VD   140 K=VAL(CHR$(KEY))
AB   150 TEXT 120,183,K
TF   160 GOSUB 240
VT   165 SOUND 0,50,10,10:PAUSE 3:SOUND
ZJ   170 IF KEY=126 THEN TEXT 120,183," ":G
     OTO 120
RH   180 IF KEY=155 THEN 260
RF   190 K2=VAL(CHR$(KEY))
NY   200 TEXT 128,183,K2
GO   210 GET KEY:IF KEY<>155 AND KEY<>126 T
     HEN 210
VK   215 SOUND 0,50,10,10:PAUSE 3:SOUND
OK   220 IF KEY=126 THEN TEXT 128,183," ":G
     OTO 160
RU   230 IF KEY=155 THEN 270
BH   240 GET KEY:IF KEY<48 OR KEY>57 AND KE
     Y<>155 AND KEY<>126 THEN 240
ZI   250 RETURN
WZ   260 NR=K:GOTO 280
UA   270 NR=K*10+K2
KV   280 IF NR>ML THEN 120
FY   290 FOR R=1 TO IS
GR   295 IF NR=T(R) THEN POP :GOTO 120
JF   300 NEXT R
FO   310 T(P)=NR
FR   320 EXEC KRZYZYK
IR   330 NEXT P
GB   335 COLOR 2:TEXT 12,183,"   LOSOWANIE
     !  ":PAUSE 50
EZ   350 FOR P=1 TO IS
VP   360 NR=1+INT(RND*ML)
FV   370 FOR R=1 TO IS
KB   375 IF NR=K(R) THEN POP :GOTO 360
JV   380 NEXT R
CJ   390 K(P)=NR:EXEC KOLKO:PAUSE 20
IM   400 NEXT P
ES   410 FOR P=1 TO IS
FM   420 FOR R=1 TO IS
ZE   430 IF T(R)=K(P) THEN IT=IT+1
JO   440 NEXT R
JJ   445 NEXT P
PZ   450 COLOR 2:TEXT 34,183," TRAFIEd: ":T
     EXT 114,183,IT:POKE 19,0
RU   452 IF IT<3 THEN EXEC S03
EJ   454 IF IT>=3 THEN EXEC S04
VY   460 DO
ET   470 IF PEEK(53279)=6 THEN 500
CO   480 IF PEEK(19)=6 THEN TEXT 8,82,P$:TE
     XT 8,90,"START - NOWY KUPON":TEXT 8,98
     ,P$
RP   490 LOOP
JW   500 POP :GOTO 75
BG   1000 PROC KRZYZYK
TF   1010 EXEC ZAL
XJ   1020 X=7+X*16:Y=-21+Y*24
RF   1025 IF P=7 THEN COLOR 1
KK   1027 EXEC S01
TM   1030 PLOT X,Y:DRAWTO X+16,Y+24
VJ   1040 PLOT X+16,Y:DRAWTO X,Y+24
YR   1050 ENDPROC
YR   1500 PROC KOLKO
TP   1510 EXEC ZAL
FV   1520 X=15+X*16:Y=-9+Y*24
KZ   1525 EXEC S02
EB   1530 COLOR 3:CIRCLE X,Y,10,16
YY   1540 ENDPROC
XD   2000 PROC ZAL
FO   2010 IF NR<=7 THEN 2100
KH   2020 Y=NR
PP   2030 REPEAT
WG   2040 Y=Y-7
JL   2050 UNTIL Y<=7
PY   2060 REPEAT
RG   2070 X=(NR-Y)/7+1
JI   2080 UNTIL X<=7
ZE   2090 ENDPROC
SI   2100 Y=NR:X=1
YI   2110 ENDPROC
LR   5000 PROC KUPON
TV   5010 GRAPHICS 31:POKE 559,0:SETCOLOR 4
     ,7,4:SETCOLOR 0,0,14:SETCOLOR 1,0,0:SE
     TCOLOR 2,3,2:COLOR 1:POKE 756,156
NR   5020 FOR R=23 TO 145-PL STEP 16
EJ   5030 PLOT R,3:DRAWTO R,170
JA   5040 NEXT R
RY   5050 FOR R=3 TO 178 STEP 24
KZ   5060 PLOT 23,R:DRAWTO 135-PL,R
JJ   5070 NEXT R
LJ   5080 FOR R=12 TO 158 STEP 24:TEXT 28,R
     ,(R-12)/24+1:NEXT R:TEXT 44,12,"8":TEX
     T 44,36,"9"
CJ   5090 FOR R=60 TO 168 STEP 24:TEXT 40,R
     ,"1":TEXT 47,R,(R-60)/24:NEXT R
CR   5100 FOR R=12 TO 120 STEP 24:TEXT 56,R
     ,"1":TEXT 63,R,(R-12)/24+5:NEXT R
HI   5110 FOR R=132 TO 160 STEP 24:TEXT 56,
     R,"2":TEXT 63,R,(R-132)/24:NEXT R
XX   5120 FOR R=12 TO 158 STEP 24:TEXT 72,R
     ,"2":TEXT 79,R,(R-12)/24+2:NEXT R:TEXT
      88,12,"2":TEXT 95,12,"9"
JJ   5130 FOR R=36 TO 168 STEP 24:TEXT 88,R
     ,"3":TEXT 95,R,(R-36)/24:NEXT R
ZX   5140 FOR R=12 TO 96 STEP 24:TEXT 104,R
     ,"3":TEXT 111,R,(R-12)/24+6:NEXT R
IS   5150 FOR R=108 TO 158 STEP 24:TEXT 104
     ,R,"4":TEXT 111,R,(R-108)/24:NEXT R
ZR   5155 IF PL THEN 5165
GJ   5160 FOR R=12 TO 168 STEP 24:TEXT 120,
     R,"4":TEXT 127,R,(R-12)/24+3:NEXT R
ZW   5165 POKE 559,34
ZD   5170 ENDPROC
QK   10000 PROC PLANSZA
EL   10010 GRAPHICS 18:SETCOLOR 0,0,0:SETCO
     LOR 3,0,0:POKE 756,156
TK   10020 POSITION 9,3:? #6;"[\":POSITION
     9,4:? #6;"]^":POSITION 6,6:? #6;"SOFTW
     ARE"
TR   10030 DL=DPEEK(560):POKE DL+13,6:POSIT
     ION 5,8:? #6;"PREZENTUJE"
SR   10040 PAUSE 50:FOR R=0 TO 15:PAUSE 4:S
     ETCOLOR 0,0,R:SOUND 0,R+5,12,R:NEXT R:
     PAUSE 8:DSOUND :PAUSE 150
ZD   10050 FOR R=15 TO 0 STEP -1:PAUSE 4:SE
     TCOLOR 0,0,R:SOUND 0,R+5,12,R:NEXT R:P
     AUSE 4:DSOUND :PAUSE 50
DP   10060 ENDPROC
SX   10490 PROC TYT
NW   10500 GRAPHICS 5:SETCOLOR 4,0,6:SETCOL
     OR 0,12,2:SETCOLOR 1,3,2:SETCOLOR 2,0,
     6:COLOR 1:POKE 756,156
BM   10505 DL=48026:POKE DL,6:POKE DL+1,6:P
     OKE DL+2,6
VH   10510 TEXT 12,0,"T O T O":TEXT 4,8,"L
     O T E K":? :? "SELECT - 7 Z 49":? "OPT
     ION - 5 Z 42":COLOR 2
AQ   10520 DO
EA   10530 XL=INT(RND*64):YL=16+INT(RND*16)
     :L=1+INT(RND*49)
KA   10540 TEXT XL,YL,L:SOUND 0,L+20,10,10:
     PAUSE 5:SOUND :TEXT XL,YL,"  ":PAUSE 3
HW   10545 IF PEEK(53279)=5 THEN GOSUB 1100
     0:GOTO 10570
MM   10550 IF PEEK(53279)=3 THEN GOSUB 1150
     0:GOTO 10570
JN   10560 LOOP
HH   10570 POP :ENDPROC
HL   11000 IS=7:ML=49:PL=0:RETURN
SD   11500 IS=5:ML=42:PL=16:RETURN
```

POLSKIE LITERY W Action!

Polski program powinien posługiwać się polskimi literami. Zasada ta obowiązuje niezależnie od zastosowanego języka programowania, w tym także Action!

Pierwsza z pokazanych procedur jest bardzo krótka, lecz możliwa jedynie uzyskanie małych liter. Przepisuje ona zestaw znaków z ROM do RAM i wykonuje w nim odpowiednie zmiany. Działanie jej jest bardzo proste i nie wymaga opisu. Trzeba jednak pamiętać o umieszczeniu wartości 128 w komórce 756 po każdym wywołaniu procedury Graphics.

Jeśli komuś to nie wystarczy, to druga procedura oferuje duże i małe litery oraz znaki umożliwiające tworzenie ramek (ale przypisane innym klawiszom niż oryginalnie). Po przepisaniu całego zestawu znaków umieszczane są w nim nowe dane definiujące wygląd znaków uzyskiwanych przez klawisze literowe, naciskane wraz z ‹CONTROL›.

Dwie następne procedury pokazują tworzenie okien w trybie 0. Do każdej z nich dodana jest również procedura demonstracyjna. Skompilowanie programu złożonego z procedury okna i procedury przykładu pozwala na obejrzenie działania okien. Pierwsza procedura jest stosunkowo powolna i widać, jak okno jest rozwijane i zwijane. Natomiast druga procedura jest bardzo szybka, lecz wymaga dodatkowej funkcji zamieniającej kod ATASCII na kody ekranu.

Grzegorz Sarnecki

```
; Polskie litery w Action!
; Grzegorz Sarnecki
; Copyright (c) Bajtek

; tylko male litery

PROC PolLit1()
 CARD n=[57344],m=[32768],a,b

  MoveBlock(m,n,1024)
  a=m+776 b=m+520
  MoveBlock(b,a,208)
  a=m+976 b=m+512
  MoveBlock(b,a,8)
  Poke(m+513,12)  Poke(m+514,24)
  Poke(m+515,124) Poke(m+527,3)
  Poke(m+537,12)  Poke(m+538,24)
  Poke(m+539,60)  Poke(m+559,6)
  Poke(m+611,28)  Poke(m+612,56)
  Poke(m+625,12)  Poke(m+626,24)
  Poke(m+627,124) Poke(m+633,12)
  Poke(m+634,24)  Poke(m+635,60)
  Poke(m+664,6)   Poke(m+665,12)
  Poke(m+721,24)  Poke(m+772,0)
  Poke(m+723,124)
; powtorzyc po kazdej Graphics()
  Poke(756,128)
RETURN

; pelny zestaw liter

MODULE
  BYTE litery=756,ch

PROC PolLit()
  BYTE f,g,j
  CARD chb,i
  BYTE ARRAY a=[
    '@  0 0 0 255 255 24 24 24
    'A  0 24 60 102 102 126 102 12
    'B  24 24 24 31 31 24 24 24
    'C  12 60 102 96 96 102 60 0
    'D  0 0 60 102 126 96 60 6
    'E  0 126 96 124 96 96 126 12
    'F  12 24 126 12 24 48 126 0
    'G  12 24 0 126 12 48 126 0
    'H  0 0 0 31 31 24 24 24
    'I  0 0 0 248 248 24 24 24
    'J  24 24 24 255 255 0 0 0
    'K  24 24 24 248 248 24 24 24
    'L  0 96 120 112 224 96 126 0
```

SOUND MACHINE

Jeżeli jesteś szczęśliwym (bądź nie) właścicielem ośmiobitowego Atari ze stacją dysków, interesujesz się muzyką i posiadasz aspiracje wielkiego twórcy, a do tego wpadasz w kompleksy czując, że twój talent się marnuje, przyjmij pomocną dłoń wyciągniętą do ciebie przez mr Fingera — autora programu Sound Machine.

Sound Machine jest półprofesjonalnym programem oferującym użytkownikowi szeroką gamę możliwości edycji dźwięku. Duża ilość funkcji, dostęp do efektownych obwiedni, a ponadto łatwość w obsłudze czynią Sound Machine niezwykle ciekawym narzędziem w rękach ludzi zainteresowanych muzyką.

EDYTOR

W wersji oryginalnej edytor zgłasza się po odczytaniu z dysku strony A. Praca w edytorze jest podstawową opcją programu.

Korzystać można z czterech pięciolinii; każda z nich odpowiada innemu generatorowi dźwięku. Fizyczna długość pięciolinii wynosi 1458 znaków, co wystarcza dla ok. 30—40 minutowego utworu. Rozpoznawane są nuty glissando i zwykłe o każdym przyjętym czasie trwania, który można wydłużyć o połowę stosując kropkę, dopuszczalne są również znaczniki, takie jak bemol czy krzyżyk. Edytor pozwala na stosowanie pauz (od 1/1 do 1/16), każdy zapis można rozpocząć kluczem wiolinowym i podzielić tzw. liniami taktu, zaznaczającymi koniec sekwencji lub utworu. Dostępne są więc wszystkie te funkcje, które stosuje profesjonalny kompozytor podczas pracy.

Pozostałe rozkazy edytora dotyczą sposobów odtwarzania utworu:

POKE — steruje generatorami dźwięku przełączając filtry;

RELEASE — przedłuża czas trwania dźwięku, pozwalając na wybrzmiewanie tonu;

TONE — kontroluje czystość tonu. T4 — ton bez zakłóceń;

BASE — określa natężenie (głośność) dźwięku;

VELOCITY — odpowiada za tempo gry.

Dodatkowo można używać obwiedni i rozkazów warunkowych.

SKOKI I ROZKAZY WARUNKOWE

Skoki przygotowujemy za pomocą instrukcji FLAG i LABEL, służących do wstawiania etykiet do utworu. Pozwala to na oznaczenie etykietami określonych sekwencji, które później będą warunkowo wykonane.

Rozkazy warunkowe nasuwają analogię do asemblera:

JUMP — bezwarunkowy skok do etykiety;

CALL — wywołanie sekwencji o określonej etykiecie, powrót po napotkaniu linii taktu;

U — złożony skok warunkowy do etykiety. Po wykonaniu sekwencji powrót;

L — analogicznie do **U**, ale bez powrotu.

EDYCJA OBWIEDNI

Obwiednie dźwięku pozwalają na najróżniejsze jego kształtowanie: miękkie, twarde, staccato, minisekwencje 0.75 sekundy, vibrato częstotliwości i amplitudy. Maksymalnie może być zapamietane dziesięć obwiedni.

Podstawowe elementy obwiedni to Attack, Hold i Decay. Sound Machine rozpoznaje trzy kategorie obwiedni: HOLD — zawiera tylko Attack i Hold, AHD — zawiera wszystkie elementy i są one dowolnie definiowane. Edycję można prowadzić po przejściu do SHAPE MENU za pomocą strzałek kursora i klawiszy numerycznych. Z poziomu edytora zaprojektowana obwiednia zostaje wywołana po użyciu instrukcji SHAPES X, gdzie X jest numerem obwiedni.

MOŻLIWOŚCI DODATKOWE

Na oryginalnym dysku Sound Machine strona B jest zajęta przez melodie demonstracyjne i programy pomocnicze. Korzystanie z nich jest możliwe z poziomu MOVE MENU.

INDEX — wyświetlenie katalogu dysku;

SAVE — zapisanie skomponowanej melodii na dysku;

LOAD — załadowanie zbioru z dysku;

APPEND — łączenie programów;

NEW — kasowanie zawartości pamięci;

CREATE — kompilacja melodii na język maszynowy.

Skompilowaną melodię można odtwarzać z poziomu Basic'a — służy do tego program TEST.BAS — bądź też dołączyć ją do własnego programu za pomocą pliku AUTOMATE.BAS.

Reasumując: polecam program Sound Machine wszystkim tym, którzy interesując się muzyką nie chcą kupować od razu syntezatora Yamaha. Polecam go również programistom, których umiejętności nie wystarczają im do udźwiękowienia z poziomu języka maszynowego nowo napisanej gry. Polecam go wreszcie znudzonym — praca z Sound Machine jest naprawdę świetną zabawą.

Powodzenia i wielu udanych kompozycji!

Master

```
; Okna ekranowe w Action!
; Grzegorz Sarnecki
; Copyright (c) Bajtek

; procedura 1

PROC Okno1(BYTE POINTER okno
           BYTE odx,ody,dlx,dly)
 BYTE a,b,c,d

  IF odx+dlx>39 THEN
    PrintE("x poza zakresem") RETURN
  FI
  IF ody+dly>23 THEN
    PrintE("y poza zakresem") RETURN
  FI
  Poke(752,1)
  FOR a=0 TO dly-1 DO
    FOR b=0 TO dlx-1 DO
      d=okno(a*dlx+b+1)
      c=Locate(odx+b,ody+a)
      okno(a*dlx+b+1)=c
      Position(odx+b,ody+a)
      Put(d)
    OD
  OD
  Poke(752,0)
RETURN

; przyklad dla procedury Okno1()

PROC Przyklad1()
 BYTE ARRAY okienko
 BYTE i,j
  Graphics(0)
  okienko="***********Nacisnij**dowolny
  **klawisz ***********"
  FOR i=0 TO 22 DO
    FOR j=0 TO 37 DO
      Put(12)
    OD
  OD
  Okno1(okienko,5,10,10,5)
  i=GetD(7)
  Okno1(okienko,5,10,10,5)
RETURN

; procedura 2

PROC Okno2(BYTE POINTER okno
           CARD odkad
           BYTE dlug, szer, skok)
 BYTE a,b,d
 CARD ekran=88,c,ek

  FOR a=0 TO szer-1 DO
    ek=odkad+a*skok+ekran
    FOR b=0 TO dlug-1 DO
      c=a*dlug+b+1
      d=okno(c)
      okno(c)=Peek(ek+b)
      Poke(ek+b,d)
    OD
  OD
RETURN

; procedura pomocnicza dla Okno2()

PROC Zamiana(BYTE POINTER okno)
 CARD i

  FOR i=1 TO okno(0) DO
    IF okno(i)<32 OR (okno(i)>127 AND
      okno(i)<160) THEN
      okno(i)==+64
    ELSEIF (okno(i)>31 AND okno(i)<96)
           OR (okno(i)>159 AND okno(i)
      <224) THEN
      okno(i)==-32
    FI
  OD
RETURN

; przyklad dla procedury Okno2()

PROC Przyklad2()
 BYTE ARRAY okienko
 BYTE i

  Graphics(0)
  okienko="******Ala*******"
  Put(125) PutE()
  Zamiana(okienko)
  Okno2(okienko,220,5,3,40)
  i=GetD(7)
  Okno2(okienko,220,5,3,40)
RETURN
```

W KYAN PASCALU PODPROGRAMY ADRES ORAZ DIR

W poniższym artykule chciałbym przedstawić sposób realizacji podprogramów uruchomionych w KYAN PASCALU, których brak dał mi się najbardziej odczuć w realizowanych programach.

Funkcja ADRES pozwala wyliczyć adres dowolnej zmiennej zadeklarowanej na zewnątrz tej funkcji. Treść funkcji jest ilustracją sposobu komunikowania się części programu zredagowanego w języku maszynowym z otoczenie napisanym w KYAN PASCALU.

Kompilator w trakcie wykonywania programu rezerwuje spójny obszar pamięci przeznaczony dla zmiennych wymienionych w częściach deklaracyjnych wykonywanych bloków. Jest to tak zwany stos arytmetyczny. Zaczyna się on od adresu 37885 i rośnie w kierunku malejących adresów. Miejsca na zmienne rezerwowane są zgodnie z kolejnością ich deklarowania w wykonywanych blokach i zwalniane w momencie wychodzenia z bloku. Dostęp do stosu arytmetycznego możliwy jest dzięki zdefiniowanej w kompilatorze etykiecie SP. Etykieta ta jest równoważna komórce 130 ze strony zerowej. Wskaźnik zawarty w komórkach SP oraz SP+1, czyli 130 i 131, wskazuje na komórkę o trzy komórki poniżej aktualnego końca stosu arytmetycznego. Takie zdefiniowanie wskaźnika stosu umożliwia, dzięki adresowaniu pośredniemu, indeksowemu względem rejestru Y, dostęp do każdej zadeklarowanej w programie zmiennej.

Typ funkcji ADRES został określony jako real, aby jej wartość mogła się zawierać w granicach od 0 do 65535. Pamięć w KYAN PASCALU numerowana jest od 0 do 32767 i dalej od -32767 do -1. Argumentem funkcji ADRES jest zmienna dowolnego typu. Zastosowano przekazanie parametru przez zmienną. W takim przypadku prawdziwym argumentem funkcji ADRES jest nie wartość zmiennej, ale wskaźnik do tej zmiennej. Zadaniem funkcji jest zamiana wskaźnika na adres. Zmienna lokalna POM służy do przechowywania wartości tego wskaźnika.

W momencie wywołania funkcji ADRES na stos arytmetyczny odkładane są w kolejności następujące zmienne: wskaźnik do zmiennej ZMIENNA (2 bajty), na razie nieokreślona wartość funkcji ADRES (8 bajtów), na razie nieokreślona wartość zmiennej POM (2 bajty). Z bilansu wynika, że zmienna POM zawarta jest w komórkach (SP)+3 oraz (SP)+4, a wskaźnik do zmiennej ZMIENNA w komórkach (SP)+13 oraz (SP)+14. Zapis (SP) rozumiany jest jako liczba typu integer zawarta w komórkach SP oraz SP+1, czyli 130 oraz 131.

Działanie funkcji ADRES polega na przepisaniu wartości komórek (SP)+13 do (SP)+3 oraz (SP)+14 do (SP)+4 oraz na konwersji zmiennej POM do typu real z uwzględnieniem sposobu numerowania komórek.

Procedura DIR służy do odczytu zawartości dyskietki. Zadeklarowane zmienne: **Nazwa** jest buforem przepływu informacji, NUMER przechowuje numer używanego IOCB pomnożony przez 16. Długość zmiennej NAZWA określono na 18 znaków, aby wyświetlane były również długości zbiorów.

Działanie procedury polega na znalezieniu zamkniętego IOCB, na wypełnieniu go odpowiednimi wartościami, na wypełnieniu bufora napisem „Dn:*.* " i na otwarciu IOCB. Po tym następuje ciąg operacji GET **record** z równoczesnym wyświetleniem pobranej nazwy. Zastosowano procedurę WRITE, ponieważ pobierane nazwy kończą się znakiem przejścia do nowej linii.

Pobieranie i wyświetlanie nazw kończy się, gdy jej pierwszym elementem jest cyfra, co jest równoznaczne z napisem o pozostałych wolnych sektorach na dyskietce.

Procedura nie jest odporna na błąd w przypadku gdy IOCB od # 1 do # 5 są zajęte. Ciągi instrukcji TXA, PHA,..., PLA, TAX są niezbędne do zachowania wartości rejestru X, którego normalnie używa procesor. Procedura wydłuża kod wynikowy programu o 5 sektorów.

Ryszard Wiech

LISTING 1

```
(* Adres zmiennej
   Ryszard Wiech
   Copyright (c) Bajtek *)

(* procedura wymaga
   wczesniejszej definicji
   TYPE typ=....          *)

FUNCTION adres(VAR zmienna:typ):real;
  VAR
    pom:integer;
  BEGIN
    BEGIN
#A
      LDY #13
      LDA (SP),Y
      LDY #3
      STA (SP),Y
      LDY #14
      LDA (SP),Y
      LDY #4
      STA (SP),Y
#
    END;
    IF pom<0
      THEN
        adres:=65536+pom
      ELSE
        adres:=pom
  END;
```

LISTING 2

```
(* Odczyt katalogu
   Ryszard Wiech
   Copyright (c) Bajtek *)

PROCEDURE dir(drive:integer);
  VAR
    nazwa:ARRAY[1..18] OF char;
    numer:char;
  BEGIN
    nazwa:='D1:*.*              ';
    nazwa[2]:=chr(drive);
    BEGIN
#A
      TXA
      PHA
      CLD
      LDX #16          ; zacznij od IOCB #1
SZUK  LDA #12
      CMP 834,X        ; czy zamkniety?
      BEQ JUZ          ; tak
      TXA              ; nie
      CLC
      ADC #16          ; nastepny IOCB
      TAX
      CMP #80          ; az do IOCB #5
      BNE SZUK
JUZ   TXA              ; rejestr X do
      LDY #3           ; zmiennej "numer"
      STA (SP),Y
      CLC
      LDA SP           ; wskaznik stosu
      ADC #4           ; powieksz o 4
      STA 836,X        ; i do adresu bufora
      LDA SP+1
      ADC #0
      STA 837,X
      LDA #18          ; dlugosc bufora
      STA 840,X
      LDA #0
      STA 841,X
      LDA #3           ; OPEN
      STA 834,X
      LDA #6           ; DIR
      STA 842,X
      JSR 58454        ; CIOV
      LDA #5           ; GET RECORD
      STA 834,X
      PLA
      TAX
#
    END;
    REPEAT
      BEGIN
#A
      TXA
      PHA
      LDY #3           ; zmienna "numer"
      LDA (SP),Y       ; do rejestru X
      TAX
      JSR 58454        ; CIOV
      PLA
      TAX
#
      END;
      write(nazwa)
    UNTIL (nazwa[1]>=chr(48)) AND
(nazwa[1]<=chr(57));
    BEGIN
#A
      TXA
      PHA
      LDY #3           ; zmienna "numer"
      LDA (SP),Y       ; do rejestru X
      TAX
      LDA #12          ; CLOSE
      STA 834,X
      JSR 58454        ; CIOV
      PLA
      TAX
#
    END
  END;
```

ZX SPECTRUM 80 KB

Istnieje wiele programów (głównie gier), które na ZX SPECTRUM 128K uruchamiają piękną muzykę i bogate efekty dźwiękowe, wytwarzane poprzez trójkanałowy generator AY. Te same programy wczytane do zwykłego "gumiaka" lub "plusa" wytwarzają jedynie nędzne piski poprzez wbudowany głośniczek, mimo że do złącza krawędziowego dołączona jest przystawka "SOUND" z układem AY.

Co jest tego przyczyną? Po pierwsze, tylko nieliczne programy rozpoznają fakt wgrania ich do SPECTRUM 128K poprzez badanie obecności układu AY. Pozostałe programy, stwierdziwszy, że mają do czynienia ze zwykłym SPECTRUM, automatycznie zakładają brak generatora AY (pamiętajmy, że przystawka "SOUND" jest naszym wynalazkiem) i nie uruchamiają procedur przeznaczonych dla AY-greka. Po drugie, ze względu na ograniczoną pamięć w SPECTRUM programiści najczęściej umieszczają fragmenty kodu programu dotyczące generatora w oddzielnym bloku, wczytywanym do któregoś z banków pamięci w SPECTRUM 128K.

Cóż więc zrobić? Przeróbka zwykłej "Spectrumny" na 128-kę jest możliwa, ale jej wykonanie nawet przez dosyć zaawansowanego majsterkowicza-elektronika jest wykluczone. Poza tym jest to rozwiązanie dosyć kosztowne. Istnieje rozwiązanie tanie i prostsze sprzętowo, jednakże wymagające umiejętności programistycznych. Jest nim dołączenie do SPECTRUM dodatkowego banku pamięci 32KB, co w sumie daje 48K+ 32K=80KB.

Przeróbka taka jest w miarę nieskomplikowana ze względu na fakt, iż SPECTRUM został zaprojektowany tak, aby najtańszy model mógł być wyposażony tylko w 16KB pamięci. Pozostałe 32KB można było dokupić i zainstalować później. W zasadzie nie produkuje się układów pamięci 32 Kbity, jednak na rynku pojawiły się tanie pamięci o takiej pojemności, będące w istocie uszkodzonymi układami 64 Kbit, w których wykryto błędy w jednej "połówce" matrycy. Znany z oszczędności pan Clive Sinclair postanowił wykorzystać je do swojej maszynki. Fakt, że w SPECTRUM mamy do czynienia z "połówkami" 64-ek nasuwa pomysł, aby zamiast nich wlutować pełnosprawne pamięci 64KB i — poprzez przełączanie tych połówek mieć dostęp do dwóch banków po 32KB.

Opisywana tu przeróbka jest rozwinięciem tego pomysłu. W celu umożliwienia programowego przełączania banków zaprojektowano port wyjściowy o adresie 255 (szesnastkowo FF). Bank pierwszy włącza się instrukcją **OUT (255),0,** drugi zaś — **OUT (255),32.**

Istnieje już wiele programów przerobionych według tego standardu (wielu takich przeróbek dokonał red. M. Pietraś — BRØMBA). Programy te pracują na SPECTRUM z pamięcią 80KB dokładnie tak, jak na 128-ce; przede wszystkim wytwarzają muzykę i efekty dźwiękowe na AY-greku. Przeróbki takie wymagają dużych umiejętności programistyczno-hackerskich, ale efekt wart jest starań.

My zajmiemy się stroną sprzętową zagadnienia. Na wstępie należy zaznaczyć, że przeróbki komputera powinny podejmować się tylko osoby, które mają już doświadczenie w konstruowaniu układów elektronicznych i znają zasady obchodzenia się z precyzyjnymi płytkami drukowanymi. Ponieważ ingerujemy we wnętrze komputera, każdy błąd, każ-

de nieostrożne machnięcie lutownicą może spowodować zwarcie na płytce lub np. przerwę w metalizacji otworów, co jest bardzo trudne do wykrycia i może spowodować poważne uszkodzenie komputera. Przede wszystkim pamiętajmy, by wszelkich manipulacji dokonywać przy **odłączonym zasilaniu.**

Pierwszym krokiem będzie otwarcie obudowy i przyjrzenie się układom scalonym pamięci 32K. Znajdują się w dwóch rzędach pod mikroprocesorem **Z80.** Jeżeli na pamięciach znajduje się napis **TMS4532** lub **M3732,** to mamy do czynienia z pamięciami 32K. Czasem może się zdarzyć, że na płytce zamontowane są już układy 64K (był moment, gdy pamięci 32K były niedostępne). W większości przypadków będziemy jednak zmuszeni dokonać zakupu układów pamięci dynamicznych 64K. Podstawowa nazwa tych układów brzmi **4164.** Spotyka się jednak różne oznaczenia: **3764, 4564, 8264, 4864** itp. Potrzebne będą nam pamięci o czasie dostępu niższym, niż 200 nanosekund i **koniecznie** z 7-bitowym cyklem odświeżania (zdarzają się — bardzo rzadko — pamięci z 8-bitowym odświeżaniem). Poza pamięciami potrzebne nam będą następujące elementy:

* Układ cyfrowy 74LS30
* Układ cyfrowy 74LS32
* Układ cyfrowy 74LS74 (może być również zwykły 7474)
* Dwie diody świecące o różnych barwach
* Tranzystor PNP, np. typu BC 158, BC 308 itp.
* Ewentualny wyłącznik (do blokowania przełączania banków)
* Kilka elementów biernych według rysunku
* Ewentualnie 8 szt. podstawek 16-nóżkowych.

Następnym krokiem będzie ostrożne wylutowanie układów pamięci 32K. Po oczyszczeniu otworów z cyny oraz sprawdzeniu, czy nie uszkodziliśmy druku lub nie powstało jakieś zwarcie na płytce, należy wlutować komplet podstawek. Jeżeli nie zakupiliśmy podstawek, wlutowujemy bezpośrednio komplet układów pamięci 64K.

Następnie należy zmontować układ przełączania banków według schematu z rys. 1. Jeżeli jest to możliwe, należy wykonać płytkę drukowaną według rys. 2. Można również zmontować układ na płytce uniwersalnej, łącząc elementy cienkim, izolowanym przewodem (np. KYNARem). Wykonaniu płytki i zmontowaniu układu należy poświęcić wiele uwagi, gdyż pomyłki mogą być bardzo kosztowne.

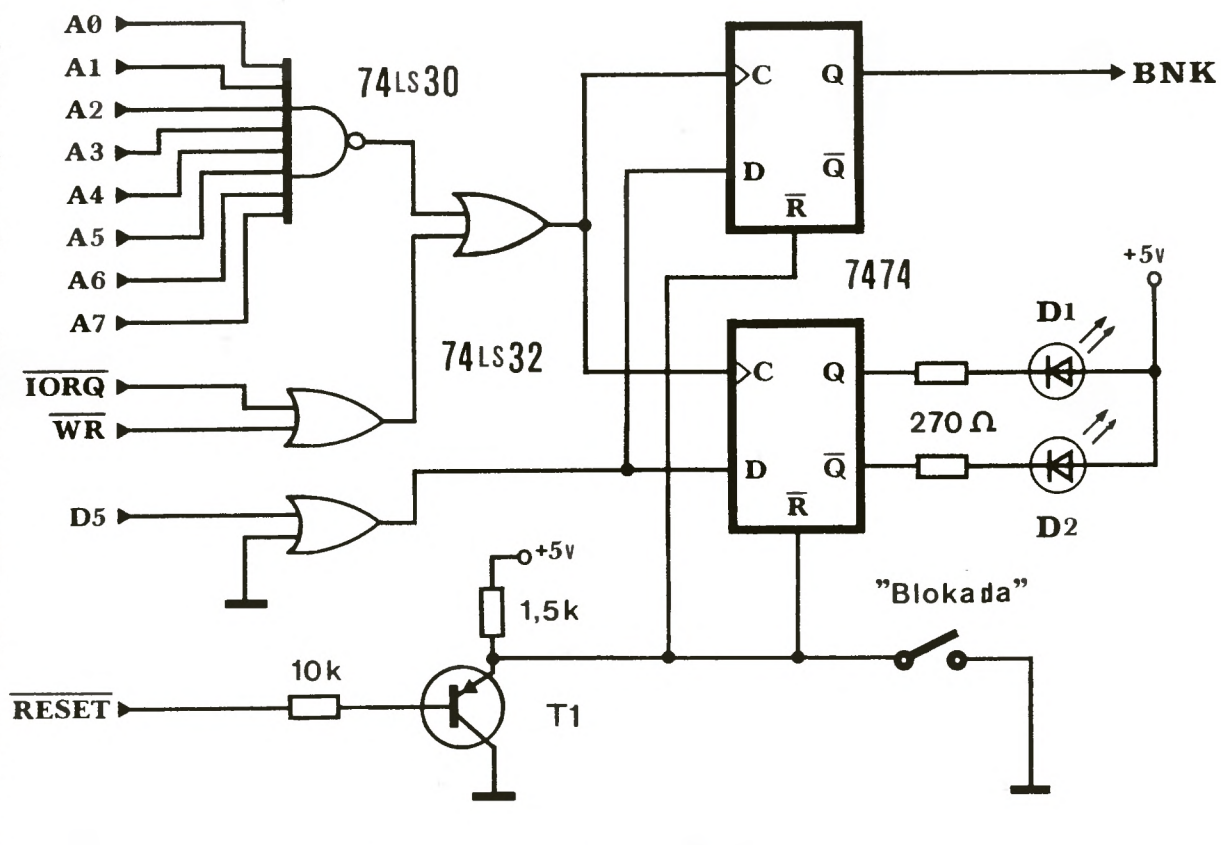
Układ ten składa się z ośmiowejściowej bramki **NAND,** dekodującej stan "1" logicznej na liniach adresowych **AO-A7,** oraz dwóch bramek **OR** dekodujących sygnały żądania dostępu do urządzenia we/wy i żądania zapisu. Powstały w ten sposób sygnał sterujący doprowadzany jest do wejścia zegarowego dwóch przerzutników "D" z układu **7474.** Do wejść danej tych przerzutników doprowadzona jest linia danych **D5** (licząc od **DO,** czyli szósta linia danych). Jest ona separowana jedną z bramek układu **74LS32,** co powoduje obciążenie szyny tylko jednym wejściem **TTL-LS** i umożliwia zastosowanie zwykłego układu **7474.** Wejścia zerujące dołączone są, poprzez wzmacniacz tranzystorowy, do wyjścia **reset,** co powoduje start komputera zawsze z aktywnym bankiem 1. Wyjście pierwszego przerzutnika steruje przełączaniem banków (sygnał **BNK**), drugi zaś przerzutnik sygnalizuje za pomocą dwu diod świecących, który bank jest w danym momencie aktywny. Wyłącznik blokujący zwiera wejścia zerujące na stałe do masy, powodując ustawienie na stałe banku 1 jako aktywnego. Wówczas SPECTRUM zachowuje się jak normalny komputer z 48KB pamięci, a polecenia przełączania banków są ignorowane.

Na rys. 2 pokazana jest płytka drukowana wraz z rozmieszczeniem elementów. Płytka charakteryzuje się dużym "upakowaniem" i nie jest zbyt elegancka, ale musi się ona zmieścić nawet w SPECTRUM "gumiaku". **M1** i **M2** są mostkami z izolowanego, cienkiego przewodu; należy je wlutować najpierw. Następnie lutujemy układy scalone i pozostałe elementy. Dwa nie oznaczone rezystory na rysunku to oczywiście rezystory szeregowe diod świecących (270 omów). Na płytce nie ma rezystora **10k** (do sygnału **reset**) — montujemy go jedną końcówką do płytki, a do drugiej doprowadzamy sygnał **reset.**

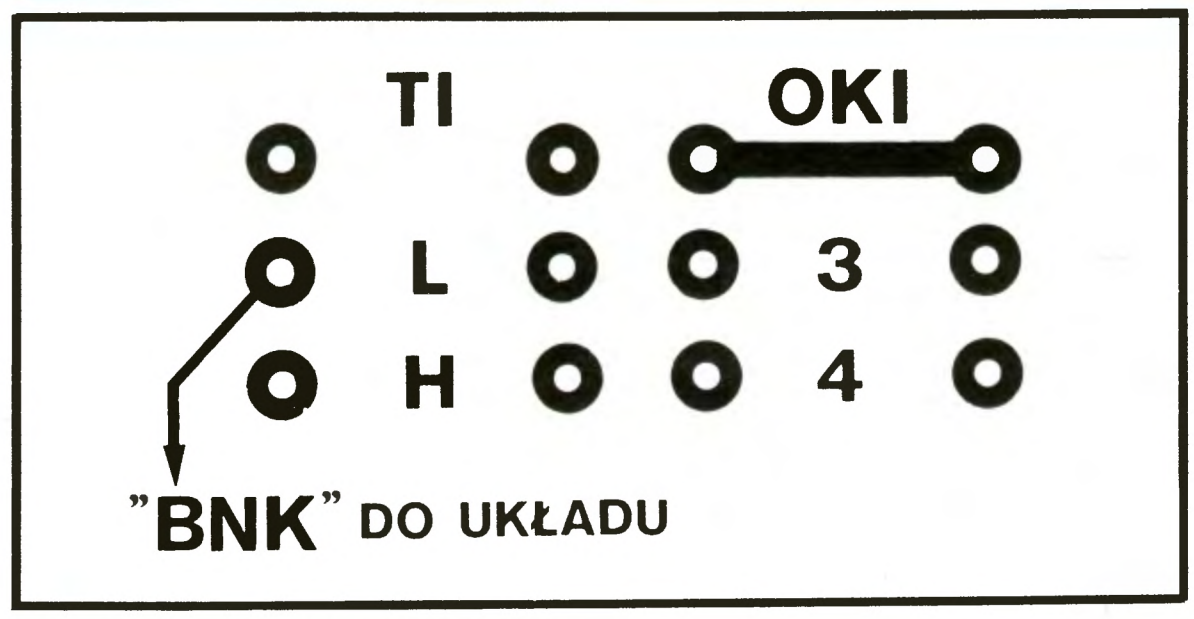
Po zmontowaniu układu i sprawdzeniu połączeń przystępujemy do zainstalowania go wewnątrz komputera. Płytka drukowana zmieści się pomiędzy układem **ULA** i pamięciami 64K. poniżej układów multipleksera adresów. Poszczególne wejścia należy połączyć z odpowiednimi punktami na płytce komputera. Najprościej dołączyć je w okolicach złącza krawędziowego — rozkład sygnałów na złączu opisany jest np. w instrukcji obsługi komputera. Połączenia wykonujemy przewodem w miarę giętkim, dającym się precyzyjnie lutować.

Osobnego omówienia wymaga sygnał **BNK,** przełączający banki, dołączany do punktu zaznaczonego na rys. 3. Na rysunku tym przedstawione są zworki znajdujące się na prawo od gniazd **EAR** i **MIC.** Są to zworki ustalające typ pamięci 32K zainstalowanych w komputerze. Jak wiemy, pamięci te są "połówkami" pamięci 64K i zworki te ustalają, która z połówek matrycy ma być wykorzystywana. Należy wylutować zworkę dolną, pozostawiając jedynie zworkę górną w położeniu prawym (oznaczonym na płytce jako **"OKI"**). Jeżeli zworka górna była wlutowana w położeniu lewym (**"TI"**), to należy ją usunąć i wlutować zworkę prawą. Następnie podkładamy kawałek materiału izolacyjnego (np. cienkiej gąbki) pod płytkę i wkładamy ją prowizorycznie na jej miejsce (węższą stroną pomiędzy pamięci 16K i 64K).

Po włożeniu pamięci do ewentualnych podstawek i po podłączeniu do układu przełączania banków wszystkich sygnałów logicznych oraz zasilania radzimy odprężyć się, przespacerować, a następnie jeszcze raz sprawdzić wszystkie połączenia. Jeśli mamy pewność, że nie popełniliśmy błędu w montażu, możemy włożyć płytkę komputera do obudowy (lub przynajmniej położyć ją na materiale izolacyjnym) i włączyć zasilanie. Jeśli komputer zgłosi się znajomym komunikatem, to możemy odetchnąć z ulgą. Jeśli nie — natychmiast wyłączamy go i spokojnie szukamy błędu w połączeniach. Maksymalna uwaga i użycie dobrych elementów powinny gwarantować sukces.



Rys. 1 Schemat układu przełączenia banków



Rys. 3 Podłączenie sygnału BNK

Jeżeli komputer nie "wybuchł" po tej próbie, przystępujemy do sprawdzenia funkcjonowania naszego usprawnienia. Przede wszystkim należy przekonać się, czy pamięć "górna" (32K) w ogóle działa. W tym celu wpisujemy polecenie **PRINT USR "a".** Jeśli w odpowiedzi otrzymamy liczbę **65368,** to OK. Jeśli jest to liczba **32600,** znaczy to, że pamięć 32K w ogóle się nie zgłasza. Powodem tego mogą być wadliwe układy pamięci lub błąd w połączeniach (w szczególności należy sprawdzić okolicę zworek obok gniazd magnetofonowych). Każda inna liczba uzyskana w wyniku powyższej próby świadczy raczej o niesprawności pamięci lub uszkodzeniu druku pod pamięciami.

Następnie ustawiamy wyłącznik blokowania układu w położeniu otwartym (układ odblokowany) i wykonujemy ciąg instrukcji:

CLEAR 32767 — przeniesienie stosu poniżej obszaru górnych 32K;
POKE 50000,121 — umieszczenie w banku pierwszym liczby 121 pod adresem 50000;
OUT 255,32 — włączenie banku 2 (zapala się dioda D2);
POKE 50000,212 — umieszczenie w banku 2 liczby 212 pod tym samym adresem, co poprzednio;
OUT 255,0 — przywrócenie banku 1 (zapala się dioda D1);
PRINT PEEK 50000 — powinniśmy otrzymać liczbę 121;
OUT 255,32 — bank 2 (zapala się dioda D2);
PRINT PEEK 50000 — powinniśmy otrzymać liczbę 212;
OUT 255,0 — bank 1 (zapala się dioda D1).

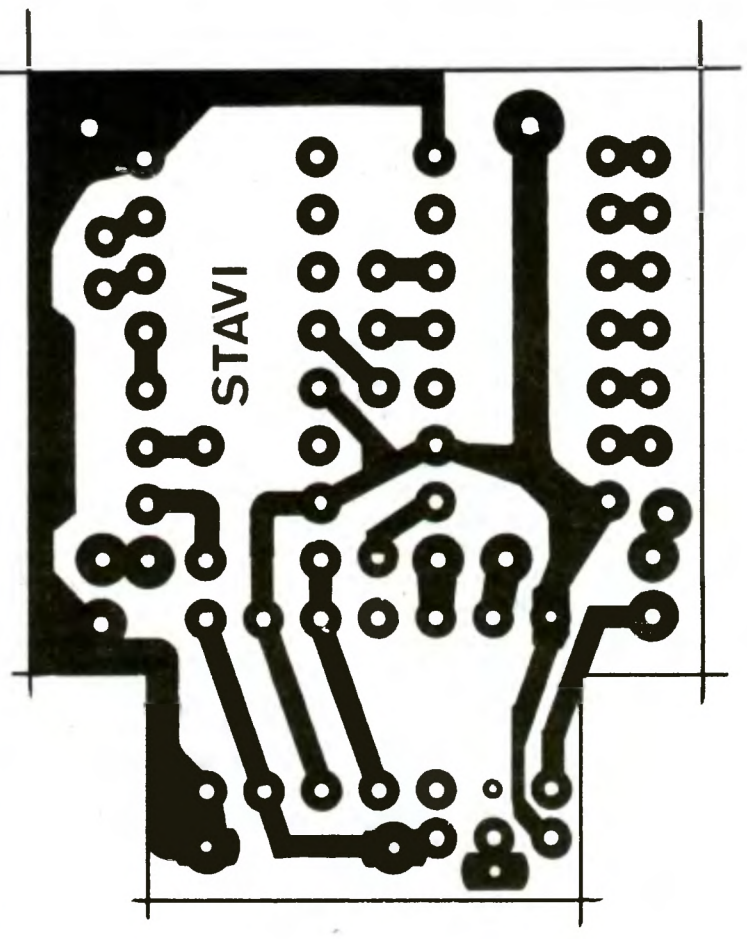
Jeżeli wszystko poszło zgodnie z opisem, to znaczy, że przeróbka działa. Zauważmy: umieściliśmy dwie różne liczby pod tym samym adresem i obie tam egzystują, znajdują się bowiem w odrębnych bankach pamięci. Bank 1 i 2 są "równoległe" w pamięci od adresu **32768,** ale w danej chwili aktywny może być tylko jeden.

Teraz pozostaje nam wkleić naszą płytkę (wraz z podkładką izolacyjną) na swoje miejsce. Najlepiej użyć do tego celu kleju typu BUTAPREN, MOMENT itp, nie przesadzając z jego ilością — w razie czego musi istnieć możliwość odklejenia płytki. Składamy komputer i... już możemy rozpocząć polowanie na programy przeznaczone dla naszego **ZX SPECTRUM 80K.** Oprócz gier, możemy również uzyskać np. program kopiujący "COPY 80", autorstwa tajemniczego pana Kato, korzystający z rozszerzonej pamięci.

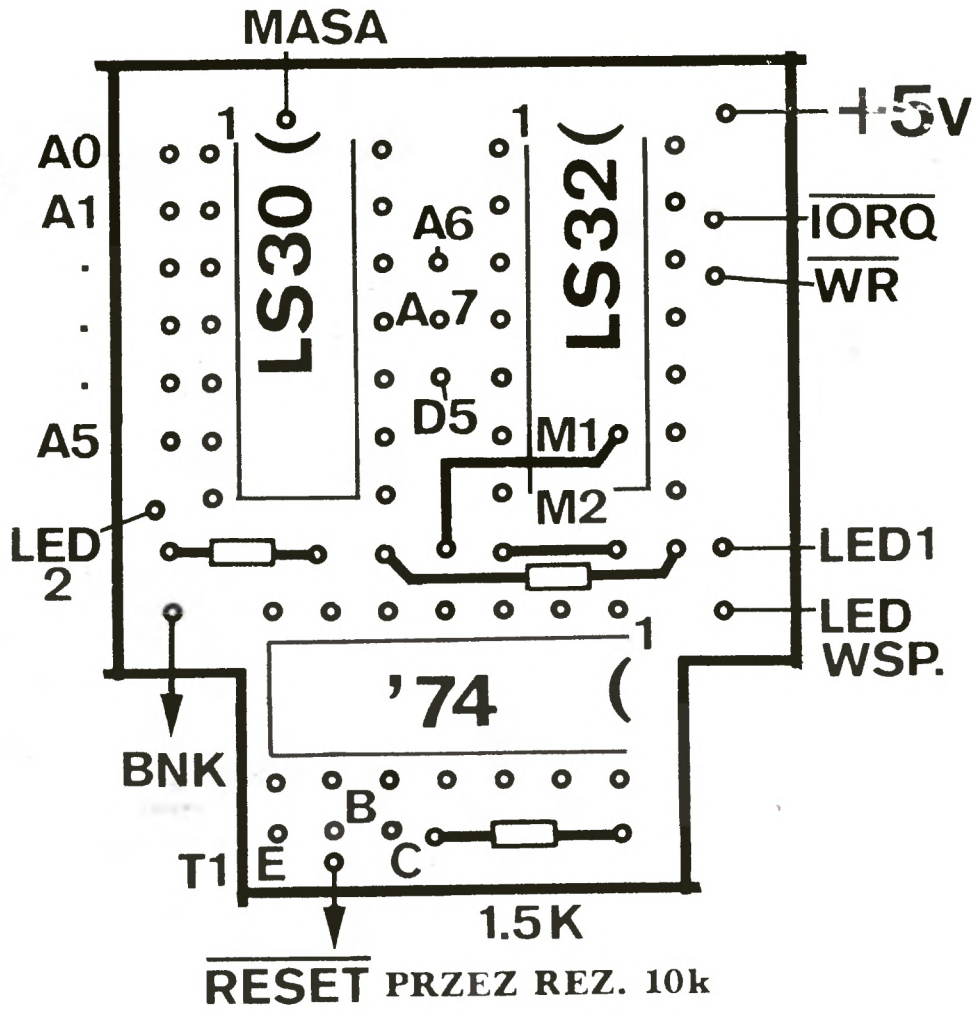
Przeróbka nie jest tak doskonała, jak przeróbka na 128K, ale jest prostsza i tańsza, a przy tym również użyteczna. Powodzenia !!!

Stanisław Winiecki

Uwaga! Schematy 2a i 2b narysowane są w skali 1:2. Jeśli chcecie z nich skorzystać, trzeba je pomniejszyć dwukrotnie (2:1), np. na ksero



Rys. 2a Rysunek płytki drukowanej

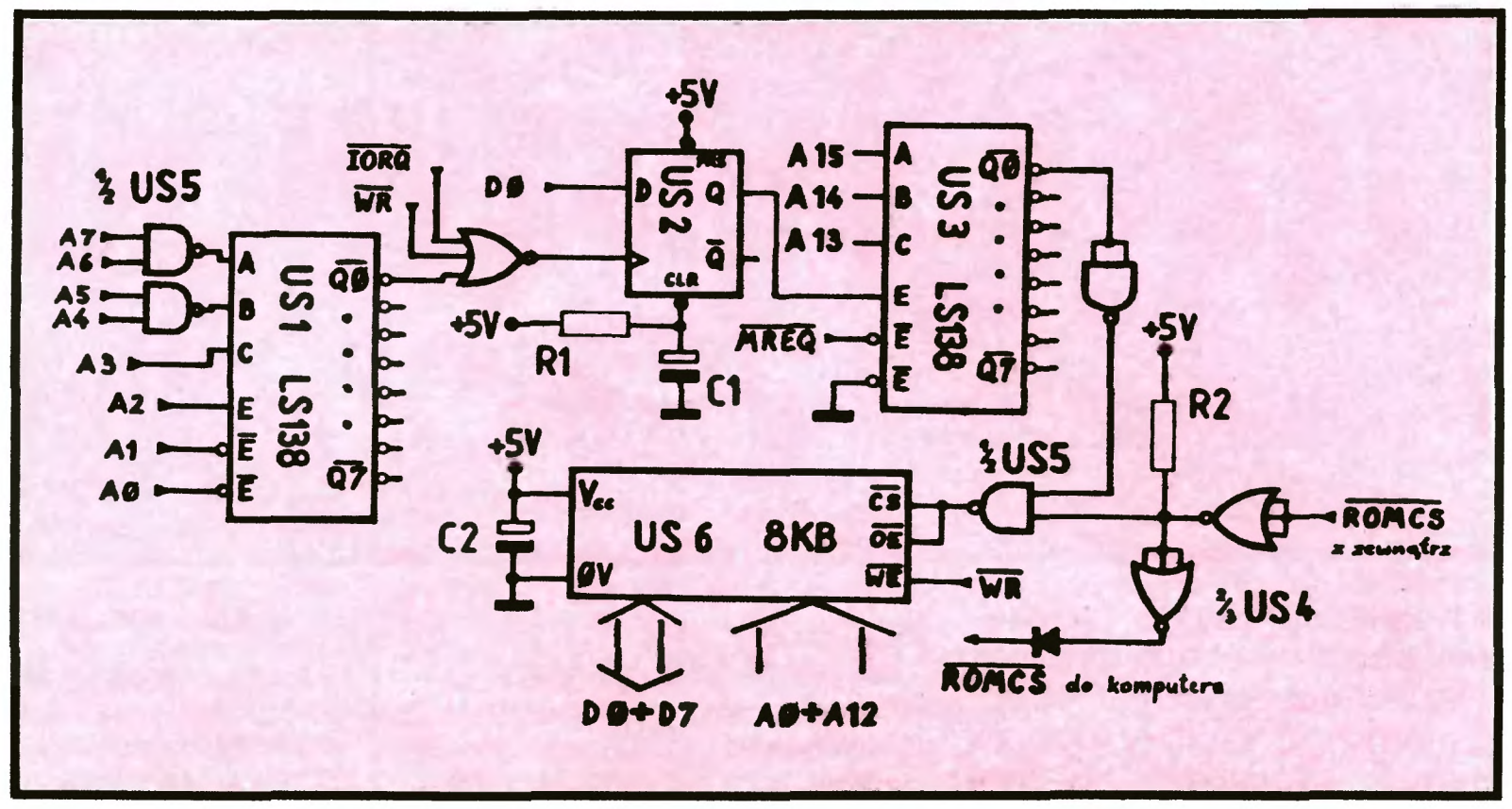


Rys. 2b Rozmieszczenie elementów i sygnałów

TIMEX

BEZ TAJEMNIC

cz. 3



W tej części omówimy najciekawszą własność Timexa — łatwą rozbudowę pamięci.

Aby przystąpić do tej rozbudowy trzeba zapoznać się z jeszcze jedną dodatkową i pożyteczną instrukcją w Timexie. Tak jak wszystkie dodatkowe instrukcje (raczej pseudoinstrukcje), można ją osiągnąć tylko poprzez porty we/wy i **tylko w Timexie!**

Timex posiada jeszcze jeden aktywny port we/wy o numerze 244, ale zanim zaczniemy go wykorzystywać do swoich celów, trzeba go przetestować rozkazem **OUT 244,1.** Jeśli w tym momencie komputer się zawiesił, na ekranie widać pionowe pasy i nie skutkuje RESET — to znaczy, że ten port jest w twoim komputerze aktywny (aby komputer znowu działał, musisz go na chwilę odłączyć od zasilania). Na przykładzie portu 244, umieszczonego we wnętrzu układu ULA, widać jak mało ma do powiedzenia mikroprocesor w Timexie.

Ten nowy pseudorozkaz spełnia bardzo ciekawą funkcję. Otóż potrafi on wyłączyć i włączyć fragment pamięci Timexa. Jego działanie opiera się na prostej zasadzie: każdy bit liczby **n** w rozkazie **OUT 244,n** steruje pracą 8 KB pamięci Timexa, np. bit nr 0 odpowiada za pierwsze 8 KB pamięci, czyli za pierwszą połówkę ROM-u, bit nr 1 — za drugą połówkę itd. Ponieważ każdy bit nadzoruje pracą 8 KB pamięci i bitów tych jest też osiem, więc razem obejmują całe 64 KB pamięci możliwej do zaadresowania przez Z 80. Każdy z bitów, gdy ma wartość 1, wyłącza bank pamięci, gdy jest wyzerowany — włącza.

Dokładne adresy banków pamięci podaje tabelka zamieszczona obok; wystarczy zsumować liczby z prawej kolumny tabeli, aby wyłączyć kilka banków naraz. Należy jednak podkreślić, że o ile można wyłączyć nawet całą pamięć, to nie jest możliwe wyłączenie mniej niż 8 KB, gdy chcemy ukryć tylko kilka bajtów. Wyłączony bank zwraca przy rozkazie **PRINT PEEK nr** same wartości 255, a co najważniejsze po ponownym dołączeniu banku pamięci mamy w całości nie zmienioną jego zawartość sprzed wyłączenia. Oprócz tego gdy bank pamięci jest odłączony, to instrukcja **POKE nr, wartość** nie potrafi nic do niego zapisać — ten bank pamięci jest naprawdę niewidoczny dla komputera i programów. Dla zapaleńców, którzy chcą wykorzystać ten pseudorozkaz, kilka rad:

— lepiej nie wyłączać banków nr 0, 1 i 2, czyli ROM-u i pamięci ekranu z oczywistych przyczyn;
— lepiej nie manipulować bankami, które zawierają stos i właśnie wykonywany program;
— przed każdym rozkazem **OUT 244** należy sprawdzić ustawienie banków rozkazem **IN 244**, ponieważ rozkaz OUT 244 dotyczy za każdym razem wszystkich banków;
— podczas pracy z bankami w BASIC-u najlepiej najpierw obniżyć **RAMTOP** rozkazem **CLEAR nr**, a potem eksperymentować;
— gdy używany jest zewnętrzny ROM, np. włączony jest interfejs POLTACT, to rozkaz **OUT 244,3** nie odłączy tej pamięci ROM, tylko jeszcze raz ROM z wnętrza komputera;
— przed eksperymentowaniem warto sprawdzić rozkazem **PRINT IN 244**, czy komputer zawsze po włączeniu zasilania ma włączone wszystkie banki pamięci (wartość zero oznacza, że tak), bo niektóre, uszkodzone Timexy mogą się źle włączać.

bit	adr. pocz.	adr. końc.	wartość
0	00000 #0000	08191 #1FFF	1
1	08192 #2000	16383 #3FFF	2
2	16384 #4000	24575 #5FFF	4
3	24576 #6000	32767 #7FFF	8
4	32768 #8000	40959 #9FFF	16
5	40960 #A000	49151 #BFFF	32
6	49152 #C000	57343 #DFFF	64
7	57344 #E000	65535 #FFFF	128

Powyższy pseudorozkaz może też być przyczyną zawieszania się nielicznych, „śmiecących" dziwnymi rozkazami gier. W takich grach może okazać się konieczne wyszukanie i wykasowanie wszystkich rozkazów typu **OUT 244+256*n**, gdzie **n=0..255.**

Teraz pora zająć się rozbudową pamięci Timexa. Pomimo przeciwnego efektu wykorzystany będzie w tym celu powyższy rozkaz **OUT 244.** Przedstawiony schemat ideowy układu nie jest konkretnym interfejsem typu MASTERFACE czy znaczną rozbudową pamięci, lecz tylko propozycją, którą można modyfikować, rozbudowywać i przystosować do własnych potrzeb. Dokładny schemat ideowy został zamieszczony na rysunku. Na jego podstawie, uwzględniając własne modyfikacje, należy wykonać płytkę drukowaną. Najlepiej gotową płytkę umieścić we wnętrzu Timexa, ale wiąże się to z dużą plątaniną przewodów oraz ingerencją w połączenia wewnątrz komputera (przecięcie ścieżki prowadzącej sygnał ROMCS na złącze krawędziowe i dołączenie obu końców do obwodu na płytce). Wykonanie płytki jako zewnętrzny interfejs jest wygodne, ale równocześnie kłopotliwe — kupno złącza i płytki ze ścieżkami na przelot, konieczność dołączania interfejsu bezwzględnie jako pierwszego w kolejności od komputera i zakaz dołączania tego interfesju do **ZX Spectrum**!

W skład układu wchodzi, oprócz kilku kostek w technologii TTL LS łatwych do dostania, także układ pamięci typu EPROM lub Static-RAM, od którego należałoby rozpocząć kompletowanie części. Pamięć ta (przykładowo jej pojemność tylko 8 KB) jest dołączana rozkazem **OUT 244,1** zamiast pierwszej połówki ROM-u. Rozkaz **OUT 244,0** przywraca stan wyjściowy. Można oczywiście wybrać inaczej, ale takie rozwiązanie wydaje się najbardziej kuszące — można w tej pamięci, jeśli jest to EPROM, zawrzeć poprawioną obsługę przerwań niemaskowalnych, poprawić inne błędy z ZX ROM-u, przerobić procedury LOAD i SAVE na tryb turbo albo w przypadku pamięci statycznej ładować tam, tuż po włączeniu komputera, z taśmy bądź dysku poprawiony ROM, program kopiujący do szybkiego wywołania lub monitor-disassembler. Możliwości jest wiele, trzeba tylko wybrać te najciekawsze i łatwe do zrealizowania — można przecież zbudować podobny układ nie wykorzystując własności portu 244 Timexa. Nie wspomniałem o możliwości dołączenia dynamicznej pamięci RAM z uwagi na znaczenie trudniejsze jej dołączenie: tzw. multipleksowanie adresów i zapewnienie prawidłowego odświeżania, ale jest to możliwe. Można wykorzystać inne banki, ale uwaga: podmienienie pamięci z tzw. Video-RAM (16384-24575) nie spowoduje, że ULA zacznie pobierać treść ekranu z nowej pamięci, tylko nadal będzie próbować tworzyć obraz telewizyjny z odłączonej, nieaktywnej pamięci.

Skoro namawiamy do modyfikacji, to warto objaśnić zasadę działania prezentowanego układu, mając cały czas nadzieję, że nie będzie on jedyną podstawą do zrozumienia jego funkcji, a chętni do jego realizacji będą posiadać niezbędne doświadczenie.

Demultiplekser US1 wraz z dwiema bramkami **NAND** z **US5** służą do jednoznacznego wykrywania adresu 244 na młodszej połowie szyny adresowej. Fakt ten sygnalizowany jest pojawieniem się zera logicznego na wyjściu **Q0 US1**. Sygnał ten po przejściu wraz z sygnałami **IORQ** i **WR** przez bramkę **NOR** z **US4** wskazuje na pojawienie się rozkazu **OUT 244**, jeśli na wyjściu bramki jest jedynka. Połączenie wyjścia bramki NOR z wejściem zegarowym przerzutnika typu **D (US2)** powoduje wpisanie do przerzutnika zawartości bitu nr 0 szyny danych, w momencie gdy procesor podaje na tę szynę argument rozkazu OUT 244. Przerzutnik pamięta ten bit i steruje nim poprzez **demultiplekser US3** i bramki, dołączaniem i odłączaniem dodatkowej pamięci zmieniając jej sygnały **CS** i **OE** (chip select i output enable). Dodatkowo zewnętrzny sygnał **ROMCS=1**, oznaczający pracę innego interfejsu z własną pamięcią, odłącza pamięć w naszym układzie. Para rezystor **R1** kondensator **C1** zapewnia wyzerowanie przerzutnika po włączeniu zasilania (wstępny rozkaz OUT 244,0) i jest niezbędnie konieczna. Kondensator **C2** blokuje zasilanie pamięci (konieczne dla układów typu MOS), a rezystor **R2** wymusza jedynkę logiczną, gdy nie jest podłączony następny interfejs podmieniający ZX ROM.

Zachęcamy jeśli nie do budowy tego układu, to chociaż do jego zrozumienia i ewentualnie opracowania ciekawszych wersji podobnych przystawek: bardziej wyrafinowanych w działaniu i oszczędniejszych w zużyciu części.

spis części:
US1, US3 — dwa demultipleksery — 74LS138 (lub polskie 74S405)
US2 — przerzutnik typu D — 74LS74
US4 — 3 bramki typu NOR — 74LS27
US5 — 4 bramki typu NAND — 74LS00
US6 — pamięć 8 KB × 8 bitów — EPROM 2764 lub SRAM 6264 itp.
R1 — rezystor ok. 4,7 kilooma
R2 — rezystor ok. 10 kiloomów
C1, C2 — kondensatory ok. 2,2 — 10 mikrofaradów
D1 — dioda najlepiej germanowa impulsowa małej mocy np. AAP 114

Marek Sawicki

INTERFACE MONITORA DLA SPECTRUM

Monitor jest podstawowym sprzętem używanym przy komputerze.

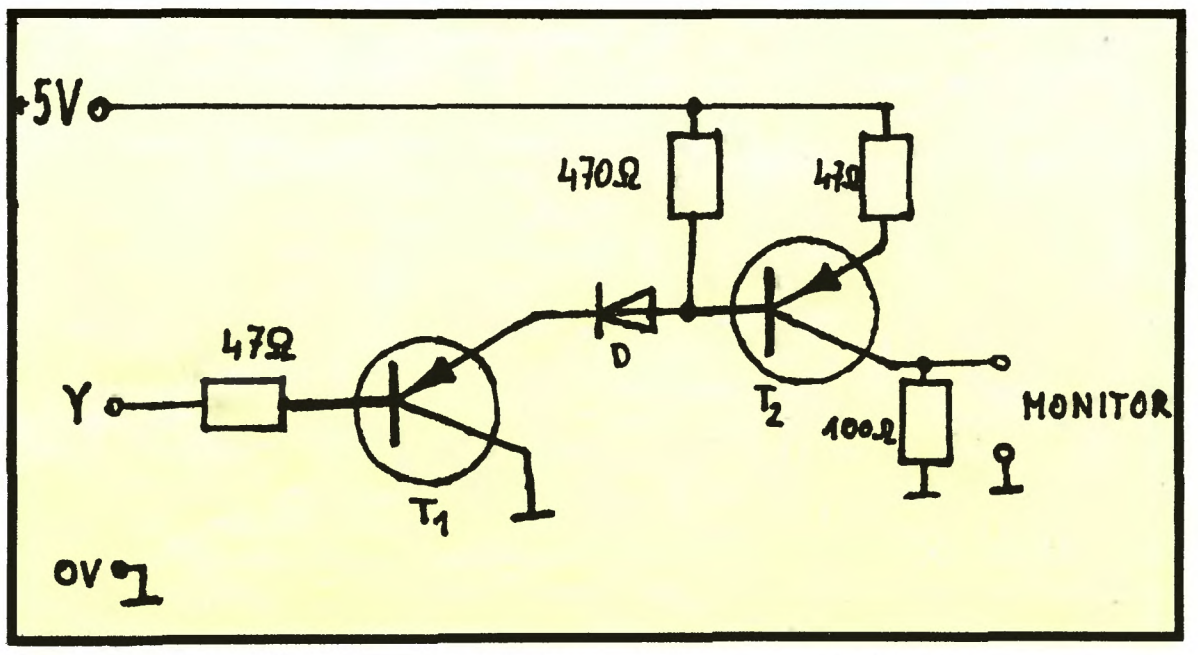
ZX Spectrum nie ma wyprowadzonego gniazda do monitora, więc połączenie tych dwu urządzeń nie jest proste. Można wykorzystać sygnał **VIDEO** dostępny na szynie krawędziowej, lecz wtedy obraz zakłócany jest przez koder kolorów.

Obok prezentujemy prosty schemat interface'u łączącego Spectrum z monitorem. Wykorzystujemy w tym celu sygnał **Y** ULI, który wzmacniamy.

Osoby mające małe doświadczenie w lutowaniu nie powinny zabierać się do montażu. Czynność tą powinni wykonać ci, którzy mają odpowiednie umiejętności, ponieważ w przeciwnym razie może dojść do uszkodzenia komputera.

UWAGA: wszystkie prace związane z przyłączaniem lub odłączaniem czegokolwiek do złącza krawędziowego powinny odbywać się wtedy, gdy komputer odłączony jest od zasilania.

Grzegorz Ostapiuk



SPIS ELEMENTÓW:

tranzystory:
BC177/179
BC157/159

rezystory:
2*47 Ω
1*470 Ω
1*100 Ω

diody:
dowolna krzemowa
np. BAP795

JĘZYK MASZYNOWY

cz. 2

Realizację cyklu artykułów pod wspólnym tytułem „Język maszynowy" rozpoczniemy od analizy dobrze znanej procedury zwanej weryfikatorem.

Zawiera ona zarówno ciekawe układy instrukcji, jak też wiele interesujących rozkazów, których poznanie przyczyni się do pogłębienia wiedzy amatorów języka wewnętrznego. Najprostszym używanym rozkazem jest rozkaz NOP, w literaturze rzadko omawiany. W procedurze został zastosowany, w związku z czym jest okazja przekazania paru uwag na jego temat.

NOP — rozkaz o kodzie 0, dla procesora oznacza dokładnie „nic nie rób". Procesorowi jest on niepotrzebny, natomiast dla niedoskonałego programisty jest ratunkiem w sytuacji, gdy w już napisanym programie dokonano kilku zmian i w rezultacie pozostało kilka wolnych komórek pamięci. Jest to puste miejsce w programie odrobinę opóźniające jego działanie. Co robi procesor napotykając kod 0? Po prostu nie wykonuje żadnych operacji i spokojnie przechodzi do następnego rozkazu.

AND — instrukcja logicznego iloczynu dwóch bajtów. Jeden bajt znajduje się zawsze w akumulatorze, drugi w rejestrze B, C, D, E, H, L (zapis rozkazu np.: AND B) albo w komórce pamięci, której adres znajduje się w rejestrze HL, IX+d lub IY+d (zapis: AND (HL), AND (IX+d)). Drugim argumentem może być także akumulator lub liczba (np. AND 15). Wynik operacji AND otrzymujemy poprzez osiem niezależnych iloczynów logicznych na poszczególnych bitach obu argumentów. Gdy oba bity są jedynkami, to w wyniku otrzymujemy bit o wartości jeden. W pozostałych trzech przypadkach ustawień bitów argumentów w wyniku dostaniemy zero. Tak skonstruowany wynik umieszczany jest w akumulatorze, a jego poprzednia zawartość ulega zniszczeniu. Wykonanie rozkazu powoduje ustawienie wszystkich znaczników w sposób zależny od wartości wyniku. Oprócz tego znacznik nadmiaru (przeniesienia) **CY** (czasem oznaczany przez C, co może mylić go z rejestrem C) jest zerowany. Stąd też częste stosowanie przez programistów rozkazu **AND A**, który nie zmienia wartości żadnego rejestru, tylko zeruje **CY** (brak jest rozkazu, który by służył tylko temu celowi).

XOR — instrukcja logicznej sumy modulo 2 (zwanej różnicą symetryczną ang. eXclusive OR). Działanie rozkazu jest podobne jak rozkazu AND, z tą różnicą, że gdy bity obu argumentów są równe (1,1 lub 0,0), to w wyniku mamy bit o wartości zero, a w przeciwnym wypadku (1,0 lub 0,1) bit wynikowy ma wartość jeden. Instrukcja XOR najczęściej jest używana w postaci XOR A, co jest najszybszym sposobem wyzerowania akumulatora. W zestawieniu z instrukcją AND i OR jest ona przydatna w obliczeniach z potęgami 2. Wynik instrukcji XOR jest umieszczany w akumulatorze.

PUSH POP — instrukcje te prawie zawsze występują parami i dotyczą operacji ze stosem. Pierwsza umieszcza zawartość pary rejestrów BC, DE, HL, AF, IX lub IY na stosie, druga zdejmuje ze stosu dwa bajty i przesyła do w/w pary rejestrów. Stosem nazywamy wydzielony obszar pamięci, traktowany jako specyficzna struktura danych, której początek (dno stosu) ma najwyższy adres, a następne dane układane są kolejno obok siebie pod coraz niższymi adresami. Adres ostatnio odłożonej danej przechowywany jest w rejestrze **SP**, zwanym wskaźnikiem stosu. Ponieważ instrukcją PUSH zawsze odkładane są dwa bajty, to ustalono, że najpierw okłada się zawartość rejestru B, D, H itd., a potem do komórki o adresie o jeden mniejszym zawartość rejestru C, E itd. i rejestr **SP** zawiera potem ten adres. Tak więc tzw. szczyt stosu zmienia się podczas każdej operacji ze stosem. Instrukcja **POP** działając przeciwnie do **PUSH** na tej samej zasadzie dokonuje odwrotnych działań: pobiera daną ze stosu do pary rejestrów i zwiększa potem zawartość **SP** o dwa. Różni te dwie instrukcje jeszcze fakt, że **PUSH** nie zmienia zawartości żadnego rejestru, ale zmienia zawartość dwóch komórek pamięci, gdy tymczasem **POP** zmienia zawartość pary rejestrów, a nie zmienia nic w pamięci. Przy użyciu stosu instrukcjami np. PUSH HL i POP BC można przenieść daną z jednej pary rejestrów do drugiej.

CP — instrukcja porównania, służąca do porównania zawartości akumulatora i argumentu tej instrukcji. Żaden rejestr nie ulega zmianie, tylko na podstawie wyniku ustawiają się wskaźniki, np. wskaźnik zera Z ma wartość 1, gdy argument i akumulator są równe. Zapis rozkazu jest identyczny jak w przypadku AND i XOR, np. CP (HL) albo CP 20;

CALL RET — rozkazy te zawsze występują parami. Pierwszy przenosi wykonanie programu w inne miejsce pamięci (wykonuje skok), gdzie po sekwencji rozkazów jest umieszczony na końcu rozkaz RET. Dokonuje on powrotu do miejsca, skąd nastąpił skok, czyli do rozkazu bezpośrednio po CALL. Skąd procesor „wie" dokąd ma wrócić? Otóż rozkaz CALL zostawia tę informację poprzez odłożenie zawartości tzw. licznika rozkazów PC na stos tuż przed wykonaniem skoku. Rozkaz RET zdejmuje ten adres ze stosu i wstawia z powrotem do rejestru **PC**. W ten sposób można w asemblerze, tak jak w innych językach, wykonywać podprogramy i procedury;

JP JR — rozkazy skoku przenoszące wykonanie programu w inne miejsce pamięci, ale bez możliwości powrotu. Oba rozkazy wykonują się identycznie, z tym że argumentem rozkazu JP jest dwubajtowy konkretny adres, pod który należy skoczyć, natomiast argumentem rozkazu JR jest liczba określająca o ile komórek w przód lub w tył ma być wykonany skok. Argument ten jest jednobajtową liczbą w kodzie uzupełnień do dwóch U2. Kierunek skoku określa najstarszy bit argumentu (0 — w przód), stąd maksymalna długość skoku wynosi 127, licząc od adresu za rozkazem JR. Taki skok nazywany jest skokiem względnym i ma zastosowanie w programach relokowalnych, tzn. takich, które można umieszczać w dowolnym miejscu pamięci.

Rozkazy **CALL**, **RET**, **JP** i **JR** oprócz zwykłej formy mają jeszcze swoje odpowiedniki warunkowe, czyli ich wykonanie jest zależne od ustawienia znaczników. Złe ustawienie powoduje zignorowanie rozkazu przez procesor.

Przejdźmy teraz do analizy programu, którego procedurę napisaną w uproszczonej formie asemblera przedstawia listing. Wszystkie instrukcje zostały ponumerowane w kolejności ich występowania w programie źródłowym. Z uwagi na brak etykiet zastosowano niekonwencjonalny sposób zapisu niektórych adresów skoków w postaci znaku „›" z podaniem liczby porządkowej rozkazu docelowego.

1. Cel programu. Każdą linię programu w BASIC-u należy uzupełnić o komentarz, w którym zostanie podana suma kontrolna linii modulo 256, tzn. reszta z dzielenia przez 256 sumy wszystkich bajtów w linii. Trzeba zatem umieścić na końcu linii dwukropek „:", słowo kluczowe **REM**, kody cyfr sumy kontrolnej i kod **ENTER**. Ubocznym efektem będzie wydłużenie się programu o 4 * liczba linii programu bajtów.

2. Istota algorytmu. Cały program w BASIC-u należy przesunąć w przód. Następnie „przerzucając" z powrotem każdą z linii na stare miejsce uzupełniać ją o kod 58 (kod „:") w miejsce 13 (kodu ENTER), 234 (kod REM), 0, 0, (miejsce na kody sumy kontrolnej) i 13 (kod ENTER). Z kolei w miejsce zer trzeba wpisać obliczoną sumę kontrolną w kodzie szestnastkowym. Do obliczenia długości linii można wykorzystać procedurę z ROM-u spod adresu 6584 (#19B8), do mnożenia przez 4 procedurę z 12457 (#30A9), a do przesunięcia programu — procedurę z 5717 (#1655).

3. Realizacja algorytmu. Całość procedury można podzielić na trzy bloki:

I. Blok przesuwający w przód o k = = ilość _ linii * k bajty. Blok ten zaczyna się od rozkazu (64) i ma pięć części:
   a. Przygotowanie danych wejściowych (rozkazy (64—68)), czyli wyzerowanie licznika linii, przygotowanie adresu końca programu (odczyt zmiennej **VARS** o adresie 23727 minus 1). Adres końca jest przechowywany w rejestrze DE, a do HL wpisuje się adres początku programu (pobierany ze zmiennej **PROG** o adresie 23635);
   b. Policzenie ilości linii (rozkazy (69—77)), wyliczenie adresu następnej linii poprzez wywołanie procedury spod adresu 6584 (wynik w DE), następnie testuje się, czy to nie była już ostatnia linia (rozkazy (72—74)) i jeśli tak, to ustawia się znacznik **CY**;
   c. Przechowanie liczby linii plus 256 (78—81) w komórkach 23728 i 23729;
   d. Pomnożenie ilości linii przez 4 (82—84), wynik mnożenia jest w HL;
   e. Przesunięcie programu w przód o HL bajtów (85—88) za pomocą procedury z adresu 5717 (w HL mamy adres początkowy, a w BC, o ile przesuwamy).

II. Blok dokonujący cofnięcia programu z dopisaniem sumy kontrolnej na końcach linii:
   a. Przygotowanie danych do przesuwania (89—92);
   b. Przerzucenie linii z powrotem pod stary adres (93—107).

Assembler programu „weryfikator" zapisany w uproszczonej formie

```
 1 NOP
 2 INC  HL
 3 AND  15
 4 ADD  A,48
 5 CP   58
 6 JR   C,>8
 7 ADD  A,7
 8 LD   (HL),A
 9 RET
10 LD   HL,(23627)
11 PUSH HL
12 LD   HL,(23635)
13 JR   >18
14 INC  HL
15 INC  HL
16 INC  HL
17 INC  HL
18 INC  HL
19 INC  HL
20 AND  A
21 POP  DE
22 SBC  HL,DE
23 ADD  HL,DE
24 JR   NC,>9
25 PUSH DE
26 LD   E,(HL)
27 INC  HL
28 LD   D,(HL)
29 DEC  HL
30 DEC  HL
31 DEC  HL
32 INC  D
33 XOR  A
34 LD   C,A
35 ADD  A,C
36 LD   C,A
37 LD   A,(HL)
38 INC  HL
39 DEC  E
40 JR   NZ,>35
41 DEC  D
42 JR   NZ,>35
43 LD   A,(HL)
44 CP   234
45 JR   NZ,>14
46 LD   A,C
47 RRCA
48 RRCA
49 RRCA
50 RRCA
51 CALL >2
52 LD   A,C
53 CALL >2
54 JR   >16
55 DEFB 58
56 DEFB 234
57 DEFB 0
58 DEFB 0
59 DEFB 13
60 NOP
61 NOP
62 NOP
63 NOP
64 LD   IX,00
65 LD   HL,(23627)
66 DEC  HL
67 EX   DE,HL
68 LD   HL,(23635)
69 PUSH DE
70 CALL 6584
71 POP  HL
72 AND  A
73 SBC  HL,DE
74 ADD  HL,DE
75 EX   DE,HL
76 INC  IX
77 JR   NC,>69
78 PUSH IX
79 POP  HL
80 INC  H
81 LD   (23728),HL
82 DEC  H
83 LD   DE,04
84 CALL 12457
85 LD   B,H
86 LD   C,L
87 LD   HL,(23635)
88 CALL 5717
89 LD   IX,23728
90 INC  DE
91 INC  HL
92 EX   DE,HL
93 INC  HL
94 INC  HL
95 LD   C,(HL)
96 INC  HL
97 LD   B,(HL)
98 INC  BC
99 INC  BC
100 INC  BC
101 INC  BC
102 LD   (HL),B
103 DEC  HL
104 LD   (HL),C
105 DEC  HL
106 DEC  HL
107 LDIR
108 PUSH HL
109 LD   BC,05
110 DEC  DE
111 LD   HL,23371
112 LDIR
113 POP  HL
114 DEC  (IX+0)
115 JR   NZ,>93
116 DEC  (IX+1)
117 JR   NZ,>93
118 JP   >10
```

Długość linii jest w BC;

c. Dopisanie na końcu przesuniętej linii kodów 58, 234, 0, 0, 13 (108—112);
d. Sprawdzenie, czy cofnięta linia nie była ostatnią (114—118). Zastosowano ciekawe rozkazy **DEC (IX+0)** i **DEC (IX+1)** oznaczające zmniejszenie o jeden zawartości komórek pamięci wskazywanej przez rejestr IX i komórki następnej, niż wskazywana przez IX. Rejestr IX wskazuje na komórkę 23728, czyli młodszy bajt licznika linii. Do chwili wyzerowania tego licznika wykonuje się w pętli cofanie i uzupełnianie kolejnych linii.

III. Blok ten oblicza i wpisuje sumę kontrolną w kodzie hexadecymalnym (2—8) i (10—55):

a. Przygotowanie danych wejściowych dla bloku (10—13) i (18—19), czyli pobranie adresu końca linii i odłożenie na stos, pobranie adresu początku linii;
b. Sprawdzenie, czy sumowana linia nie była ostatnią (20—25). Zerowany jest znacznik **CY**, pobierany ze stosu nowy adres końca programu, a po sprawdzeniu przekazuje się adres początku programu z powrotem na stos;
c. Przygotowanie licznika ilości bajtów (26—28) oraz adresu początku linii (29—31). Długość linii liczy się od jej piątego bajtu do kodu **ENTER** włącznie;
d. Sumowanie bajtów linii modulo 256 (32—42). Podczas sumowania rejestr C przechowuje sumy cząstkowe i wynik. Na koniec w HL jest adres kodu **REM** w modyfikowanej linii;
e. Sprawdzenie, czy na końcu linii istnieje suma kontrolna (43—45). Jeśli nie, to wykonuje się skok do rozkazu (14). Jest to potrzebne podczas weryfikacji programu, gdyż program weryfikowany może nie mieć w niektórych liniach sum kontrolnych;
f) Ustalenie, na podstawie zawartości rejestru C, wartości obu cyfr hexadecymalnych sumy kontrolnej. Rozkazy (46—51) ustalają bardziej znaczącą cyfrę, a (52—53) mniej znaczącą. Działanie tych rozkazów zasługuje na szczególną uwagę i omówiono je dokładnie w punkcie 4. Następnie skok (54) przenosi wykonanie programu do rozkazu (16), gdzie cały cykl powtarza się dla następnej linii. Jeżeli poprawiona została ostatnia linia, to znacznik **CY** w rozkazach (22—23) nie zostaje ustawiony, wykonuje się skok (24) i program kończy swoje działanie wracając do BASIC-a.

IV. Ciekawostki na temat niektórych zestawów rozkazów:

Do jednych z ciekawszych efektów należy zaliczyć zestaw rozkazów (20—23). Są one czujnikiem, dającym sygnał do zakończenia programu w momencie, gdy do ostatniej linii została dopisana suma kontrolna. W rejestrze DE jest adres końca programu, a w HL adres początku kolejnej linii. Po przerobieniu ostatniej linii HL wskazuje już na nieistniejącą linię poza programem i nierówność HL < DE zmienia się na HL > DE, co jest „wyłapywane" i następuje zakończenie programu.

Spójrzmy na te rozkazy:

```
20. AND A
21. SBC HL, DE
22. ADD HL, DE
23. JR NC, >9 (czyli skok do rozkazu RET)
```

Rozkaz **AND A** jest tylko po to, aby wyzerować **CY**, co jest konieczne, aby poprawnie wykonał się rozkaz **SBC**. **SBC** wykonuje działanie: HL = = HL − DE − CY. W rezultacie wykonania rozkazów (21) i (22) zawartości rejestrów HL i DE się nie zmieniają, ale wynik odejmowania ustawia odpowiednio znacznik **CY** (CY=1, gdy HL < DE). Ustawiony znacznik **CY** nie wpływa na wykonanie rozkazu **ADD**, gdyż jest to zwykłe dodawanie bez przeniesienia: HL = HL + DE. Problemem jest fakt, że rozkaz **ADD** również zmienia znacznik **CY**, ale co ciekawe, w tym przypadku jest on tak samo ustawiony po wykonaniu dodawania, jak i przed wykonaniem. Warto przeanalizować dlaczego się tak dzieje!

Drugim interesującym zestawem rozkazów jest zestaw (46—50) oraz (52). Rozkazy te zamieniają jednobajtową liczbę na dwa kody hexadecymalne tej liczby.

Zapiszmy te rozkazy:

```
46. LD A,C          52. LD A,C
47. RRCA            ----------
48. RRCA             3. AND 15
49. RRCA
50. RRCA
----------
 3. AND 15
```

Rozkazy po lewej stronie wyznaczają wartość bardziej znaczącej, a po prawej wartość mniej znaczącej cyfry postaci hexadecymalnej liczby z rejestru A. Metodę tą, po przeróbkach, można stosować również do konwersji liczb wielobajtowych.

Ustalenie bardziej znaczącej cyfry:

Rozkaz (46) pobiera do akumulatora liczbę przechowywaną w rejestrze C, natomiast cztery rozkazy **RRCA** dokonują 4-krotnego przesunięcia akumulatora w prawo. Każde przesunięcie jest równoznaczne z podzieleniem liczby przez dwa, a łącznie z podzieleniem przez 2 do potęgi 4, czyli przez 16. Zapis wyniku tego ilorazu jest na razie bardzo dziwny, gdyż rozkaz **RRCA** dzieląc równocześnie wsuwa resztę z dzielenia, za pośrednictwem znacznika **CY**, od strony najstarszego bitu akumulatora. Istotne jest, że po czterech przesunięciach interesujące nas cztery najstarsze bity akumulatora znalazły się na kolejnych mniej znaczących pozycjach, a inne bity po prostu nas nie interesują. Aby się ich pozbyć, został zastosowany rozkaz (3), który dokonuje operacji **AND** akumulatora i liczby 15. Wystarczy jeszcze zauważyć, że liczba 15 to dwójkowo 00001111, aby zrozumieć jak ten rozkaz zeruje niepotrzebne dla nas bity akumulatora, pozostawiając nietknięte cztery najmniej znaczące pozycje. Otrzymana w akumulatorze liczba jest zatem częścią całkowitą z dzielenia z początkowej zawartości akumulatora przez 16. Teraz wystarczy sprawdzić, czy jest ona większa od 10 i gdy nie, to zastąpić ją kodem **ASCII** (znakiem) cyfry, a gdy tak, to zastąpić ją znakiem litery A, B, C, D, E lub F.

Ustalenie mniej znaczącej cyfry:

Drugą cyfrę otrzymamy wyliczając resztę z dzielenia liczby przez 16. Rozkazem (52) pobieramy znowu liczbę do akumulatora i zauważamy, że ta potrzebna nam reszta z dzielenia to nic innego jak cztery najmłodsze bity w akumulatorze. Wystarczy zatem odrzucić 4 bardziej znaczące bity i otrzymujemy interesującą nas drugą cyfrę. Możemy zastosować ten sam, co poprzednio, rozkaz **AND 15** i w ten oto sposób zamieniliśmy jednobajtową liczbę na dwa znaki jej reprezentacji szestnastkowej.

Piotr Sumara

TOS inaczej

Timex Operating System — dyskowy system operacyjny — został tak pomyślany, aby był jak najbardziej przyjazny dla użytkownika.

Wszystkie rozkazy zaprojektowano w ten sposób, aby obsługa dysku była analogiczna do obsługi taśmy. Polecenia wprowadza się za pomocą słów kluczowych BASIC-a z rozszerzeniem w postaci „*" (gwiazdki). Sam system mieści się na każdej sformatowanej „pod nim" dyskietce i zajmuje niecałe 16K.

TOS posiada strukturę drzewiastą, co oznacza, że katalog może zawierać podkatalogi, które z kolei też mogą zawierać kolejne podkatalogi itd. Jest to bardzo wygodne i stanowi duży krok w kierunku takich struktur jak **MS-DOS**. Niestety, mała pojemność dyskietki sprawiła, że hierarchiczny system zbiorów jest raczej ciekawostką i nie może być w pełni wykorzystany.

System Timexa ma jeszcze jedną poważną zaletę: nie zajmuje pamięci RAM komputera. Dzieje się tak dlatego, że układ kontrolera posiada własny procesor Z-80 i 16K (albo 64K w FDD 3000) pamięci RAM, a analiza składni poleceń dla stacji odbywa się sprzętowo przez interface, który też ma własną pamięć.

Działanie TOS-u nie ogranicza się tylko do zarządzania dyskiem, system umożliwia współpracę z urządzeniami zewnętrznymi za pośrednictwem dwu łącz transmisji szeregowej RS 232.

Ta wspaniała praca portugalskich programistów doczekała się swoich przeróbek w Polsce. Taką próbą usprawnienia działania TOS-u jest praca Radosława Cymera, znanego jako Wise Man.

Przeróbce swej nadał on nazwę **„TOS VA.4"** i jest to próba ulepszenia systemu w wersji VA.2. Nowy system zapisany jest na specjalnym dysku systemowym, z którego ładuje się do pamięci stacji (wymagana jest stacja z 64K pamięci RAM). Dyski sformatowane w „nowym" systemie posiadają więcej miejsca na pliki (164K wolnej pamięci na jednej stronie) dlatego, że nie jest na nich zapisany system oraz formatowane są na 42 ścieżki po 16 sektorów każda. Aby uzyskać dostęp do tak sformatowanych dysków, należy wgrać system z dysku głównego (tego z systemem) — cały system znajdzie się w pamięci stacji i będzie się zgłaszał po każdym wyzerowaniu komputera.

Zgłoszenie następuje przez pojawienie się programu **START**, z którego możemy załadować jeden ze stale znajdujących się w pamięci stacji programów lub po prostu wgrać program z dysku. Programy stale dostępne dla użytkownika to: Compress Copy, Zebra Copy, So-So Copy, Mega Phantom, Disk editor, assembler oraz kompresory i monitory. Stanowi to ciekawe rozwiązanie, i znacznie ułatwiłoby posługiwanie się dyskami oraz pracę z komputerem, gdyby nie kilka usterek, które postaram się opisać.

Wszystkie programy zaadaptowane dla systemu TOS VA.4 są tworami nie bardzo przyjaznymi dla użytkownika i zawierają błędy. Każdy program jest pracą innego programisty i w rezultacie daje to dziwną mieszankę profesjonalizmu z dyletantyzmem. Na przykład program **START** nie daje nam możliwości poruszania się po podkatalogach, dezorientuje i wprowadza chaos, w szczególności jeśli dysk zawiera hierarchiczną strukturę plików.

Programy kopiujące stale dostępne byłyby wspaniałym rozwiązaniem, gdyby nie trudności w posługiwaniu się nimi. W rzeczywistości oznacza to szybkie zdenerwowanie użytkownika i chęć powrotu do prostych TOS-owych poleceń.

Charakter omawianego systemu jest wyraźnie roboczo-narzędziowy, co naturalnie implikuje wykorzystywanie go do pracy. Lecz tu możemy się bardzo zawieść. Niejednokrotnie utraciłam wielogodzinną pracę właśnie „dzięki" systemowi VA.4.

Kolejnym niedociągnięciem w pracy pana Cymera jest formatowanie na 42 ścieżki — nie wszystkie napędy mają możliwość przesuwania głowicy poza 40. ścieżkę i formatowanie takie jest ryzykowne.

Idea, jaką wymyślił i zastosował pan Cymer, jest bardzo dobra, jednakże wykonanie pozostawia wiele do życzenia. Zabrakło prawdopodobnie doświadczenia i pełnej znajomości oryginalnego systemu.

Do czego, w takim razie, przydatny jest twór Wise Mana? Idealne jest wykorzystywanie systemu TOS VA.4 do przechowywania gier. Zwiększona pojemność dyskietki pozwala „upchnąć" jeszcze jedną grę na dysku, a programy kopiujące ułatwią kopiowanie jej dla kolegi.

Reasumując: „nowy" system jest bardziej ciekawostką, niż narzędziem pracy, stary, dobry TOS VA.2 wciąż jest najlepszym systemem dyskowym dla Spectrum.

ZALETY SYSTEMU TOS VA.4:

— większa pojemność dyskietki
— wbudowane programy użytkowe

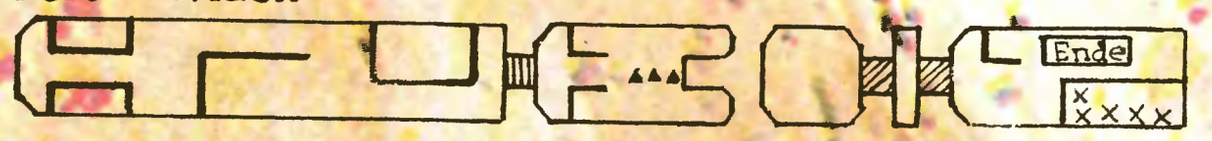
WADY SYSTEMU TOS VA.4:

— brak pracy z podkatalogami
— mało przyjazny w obsłudze
— częste skłonności do zawieszania się
— możliwość uszkodzenia przechowywanych danych
— brak ogólnej estetyki programowej

Aneta Bryske

Level 8 Tungsten

Level 9 Iridon

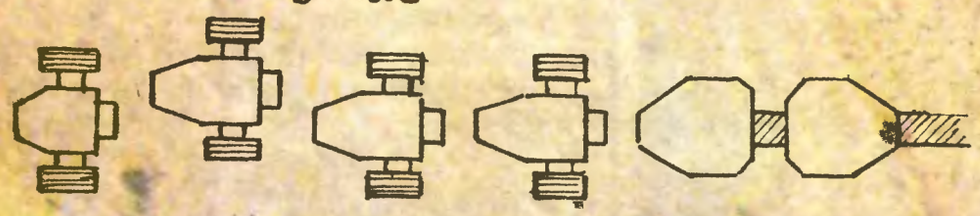


Level 10 Kallisto

Level 11 Tri-allay

Level 12 Quadmium

Level 13 Ergonite



Level 14 Galactium

Level 15 Uridium

BECIA & PEGAZ ASS

Level 1 Zine

Ende

Level 2 Lead

Ende

Level 3 Copper

Ende

Level 4 Silver

Ende

Level 5 Iron

Ende

Level 6 Gold

Ende

Level 7 Platinum

Ende

Ende

Ende

Ende

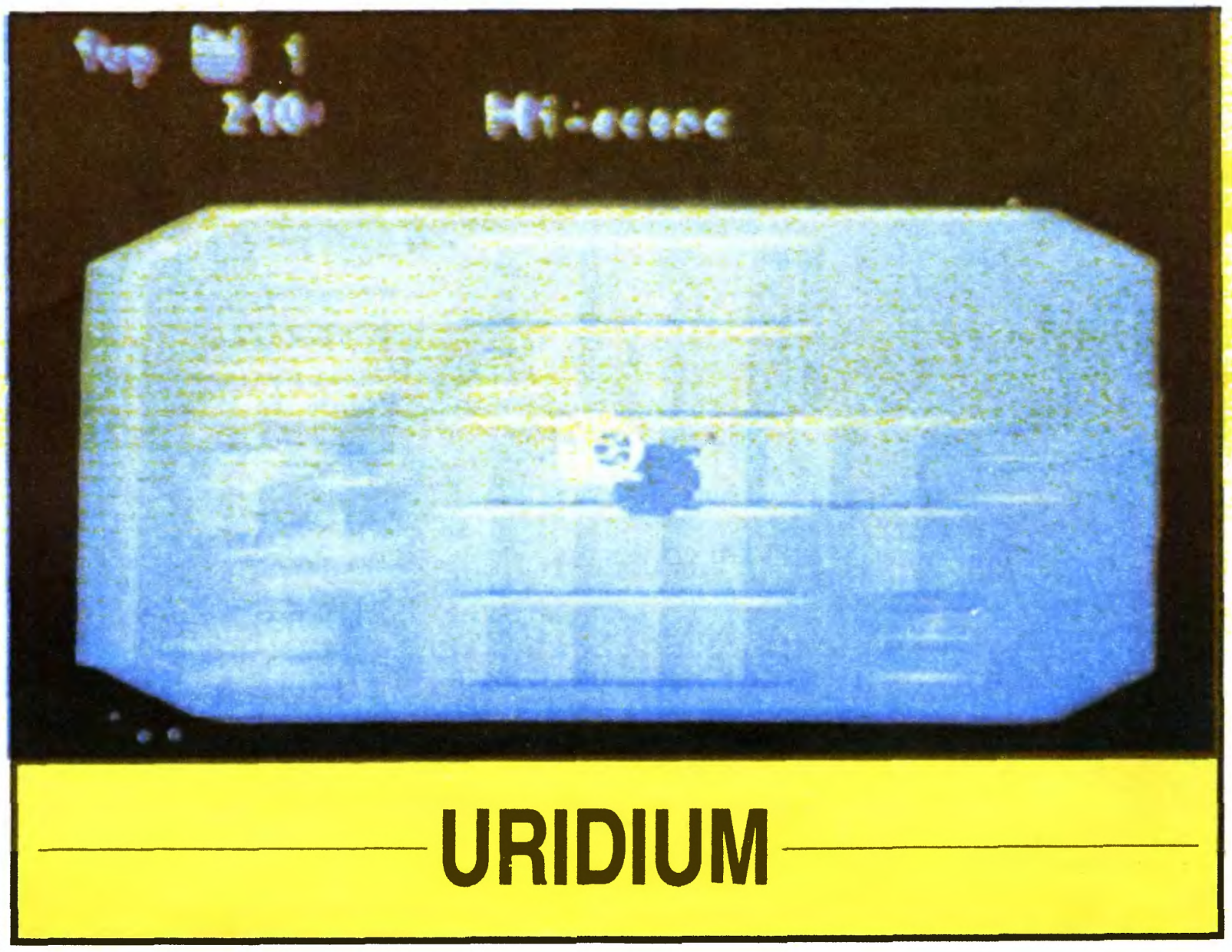
BAJTKOWA LISTA PRZEBOJÓW 9–10

Minęły wakacje, skończyły się urlopy. A nowych gier jakoś nie widać. Musimy poczekać jeszcze parę tygodni. Czy to coś da, nie wiadomo, tym bardziej, że skupisko przemytników gier — giełda na Grzybowskiej — przeżywa wstrząsy.

A więc czekamy, lecz wciąż bawiąc się starymi, dobrymi przebojami.

		ATARI	AMSTRAD	COMMODORE	SPECTRUM
1	Iron Lord	>	×	×	<
2	Fighter Bomber	>	×	×	<
3	Batman	>	<	×	×
4	The Untouchables	>	<	×	×
5	Passing Shot	>	<	×	×
6	Aliens	>	<	×	<
7	R-Type	>	<	×	<
8	Cabal	>	<	×	<
9	Total Eclipse	>		×	<
10	Robbo	<			

Kopertę nadesłał Piotr Sosiński z Wrocławia



Powstała dawno i daleko stąd. Stała się klasyką „strzelanin". Zadziwiła grafiką. Oszołomiła animacją. Obecna jest na wszystkich popularnych komputeracn. Grają w nią do dziś. Nie nudzi się, choć nie mozna powiedzieć, by była szczególnie urozmaicona.

Uridium.

Dany jest mały statek kosmiczny wyposażony w dwa równoległe działka. Napęd statku jest fotonowy, energia czerpana jest z niewielkiego reaktora. W środku znajduje się miejsce na jedną osobę i sporo elektroniki. Widok dostępny jest tylko przez przednią szybę, specjalnie wzmacnianą i zabezpieczoną przed zderzeniem z małymi ciałami.

System niewielkich dysz pozwala na lot w każdej z podstawowych płaszczyzn, oczywiście w stosunku do najbliższego skupiska materii. Statek może też wirować wokół osi podłużnej i z trudem wokół poprzecznej.

W środku, jak to było mówione, siedzi Pilot. Zadaniem Pilota jest takie kierowanie statkiem, by nie został on zestrzelony, a zestrzelił jak najwięcej innych (wrogich oczywiście) statków.

Daleko w przestrzeni kosmicznej, zawieszone w niesamowitej próżni, na falach z całego Wszechświata, kołyszą się ogromne kosmiczne krążowniki. Są nieruchome. W środku nie ma nikogo, zresztą i tak brak byłoby miejsca. Działa tylko system obronny — wykrywacze ruchu, sprzęgnięte z automatycznymi działami, czujniki podczerwieni, uruchamiające wyrzutnie samosterowanych rakiet i gotowe do startu eskadry automatycznych myśliwców gwiezdnych.

Na jednym z krańców krążownika znajduje się niewielki pas startowy, który wbrew nazwie służyć będzie do lądowania. Narysowane na tytanowej powłoce strzałki wskazują kierunek podchodzenia do lądowania.

W takiej to scenerii odbywa się granie. A polega ono na zdobywaniu kolejnych krążowników. Wszystko, co jest do zrobienia, to przelot kierowanym stateczkiem na drugą stronę krążownika i wylądowanie.

Powiedzieć — łatwo, wykonać — trudniej. Zanim doleci się na drugą stronę, przebyć trzeba wystające murki, bloki metalu i rusztowania. Na szczęście z pobliskiego Słońca pada niewielkie światło i oświetla przeszkody, które rzucają cień.

Lotnisko często jest ukryte i trzeba dobrze patrzeć, by je dostrzec. A potem wystarczy tylko zbliżyć się doń z odpowiedniego kierunku i zmniejszyć prędkość — lądowanie będzie automatyczne.

Jak bronić się przed atakami automatycznych myśliwców i całego systemu obronnego? Wystarczy prędko lecieć i odpowiednio, szybko strzelać. Same myśliwce nie są zbyt groźne, gorzej, gdy zaczynają strzelać.

Pilotowany stateczek ma dużą bezwładność, jak to w przestrzeni kosmicznej. Wyhamowanie lub przyspieszenie zajmuje więc cenny czas, podobnie jak skręty. Wymagana jest czujność i maksymalne skupienie.

W sytuacjach awaryjnych można przewrócić się na bok, lecz to nie robi dobrze żołądkowi...

Każdy krążownik nazwany jest nazwą metalu lub szlachetnego stopu. Zaczyna się od cynku, a kończy... na Uridium.

Znajomy morderca kosmitów*) chwalił się raz, że zdobył 90.000 punktów dochodząc do połowy floty...

Firma: Hewson
Komputer: Spectrum, Commodore, Amstrad, Atari, IBM

*) gryps ten ukradłem

Gen

S.O.S.

Mam Atari 65XE. Poszukuję gier COMMANDO' NINJA, BRUCE LEE, BARBARIAN i WINTER GAMES. W zamian oferuję inne gry. **Igor Zbyryt, ul. Helska 27-33 m 27 81-056 Gdynia.**

Proszę o mapy do gier SABOTEUR, COMMANDO i IKARI. Pomóżcie! **Paweł Sokół, blok 38 m 5 05-131 Zegrze Płn.**

Poszukuję gier POP EYE, CHIMERA i BARBARIAN II. W zamian 5 innych gier. Mam Atari 65XE. **Hubert Kozik, ul. Marchlewskiego 31 m 29 94-047 Łódź**

Kto mi przyśle na Atari dokładny opis gry SILENT SERVICE, informację, jak usuwa się toster w grze CHIMERA, jak przejść trzecie jeziorko w QUEST FOR TIRES? **Konrad Jackowski, ul. Czcibora 4 m 16 71-580 Szczecin**

W zamian za opis do gry THE CORE oferuję opisy do gier PTRAP DOOR i WEST BANK oraz grę PACMANIA. **Jakub Tatarczak, ul. Jasińskiego 11 20-807 Lublin.**

Liczę na waszą pomoc w zdobyciu gry MIKIE. Mogę za to ofiarować gry FAIRLIGHT, THREE WEEKS IN PARADISE i inne na komputer Timex 2048. **Krzysztof Kolanowski, ul. Skłodowskiei 25 m 456 85-088 Bydgoszcz**

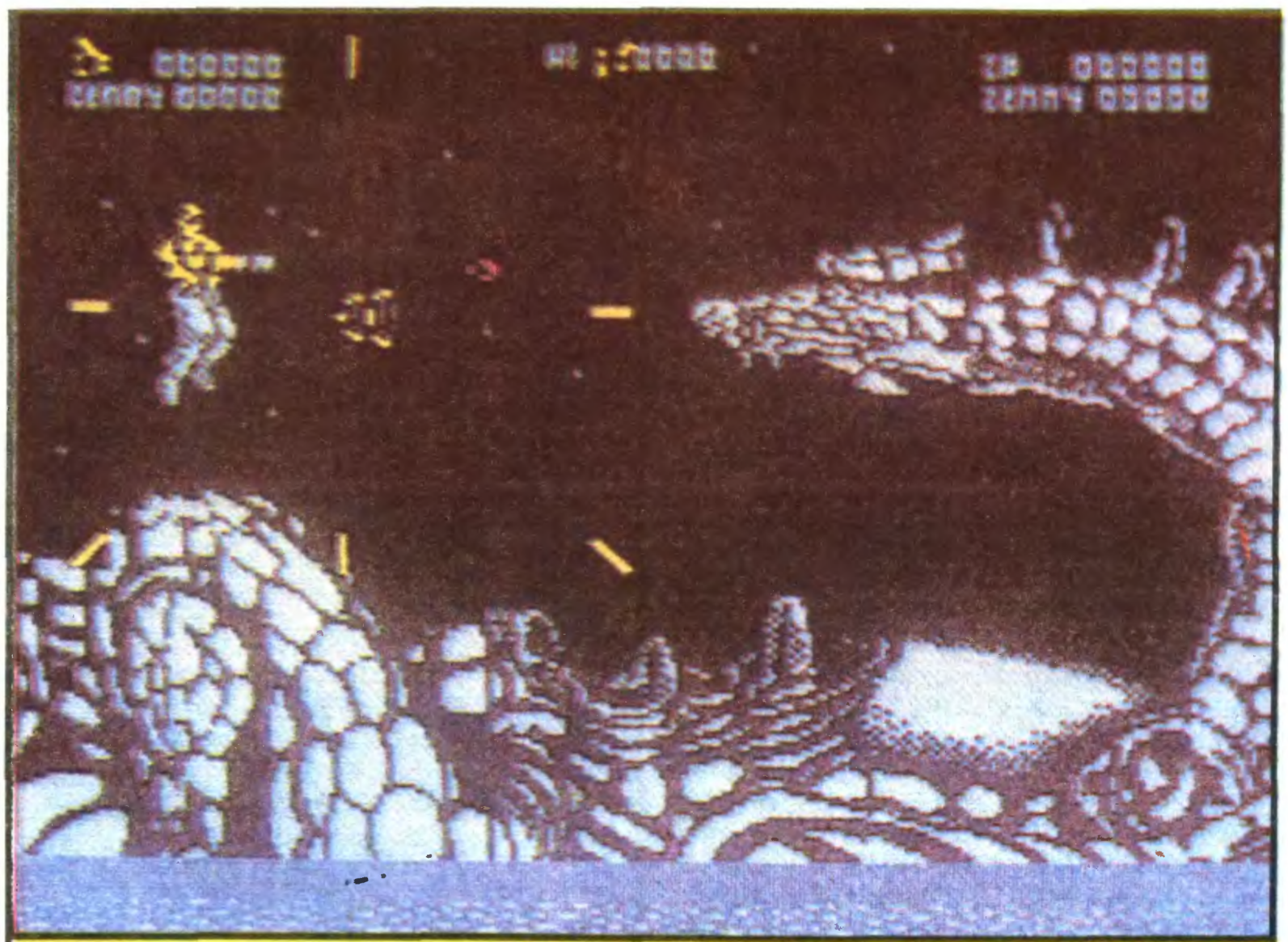
Za gry BARBARIAN i GHOST's and GOBLINS na Atari odstąpię gry ALLEY CAT, NINJA, ZORRO i GLADIATOR. Mam magnetofon i stację dysków. **Andrzej Kopeć, ul. Partyzantów 22b m 10 47-220 K-Koźle**

Od kilku tygodni bezskutecznie poszukuję gier SILENT SERVICE i THE TRAIN na Atari 65XE z magnetofonem XC12. W zamian oferuję gry TENNIS, ARCHON, GREMLINS i LASER GATES. **Artur Rynkiewicz, ul. Upalna 56 m 26 15-668 Białystok (Słoneczny Stok)**

Szukam opisów gier RED MAX, ZORRO i SABOTEUR II. Za te opisy odstąpię inne. **Paweł Zawada, ul. Bacciarellego 26 m 2.**

Potrzebuję gry BARBARIAN, GHOST CHASER, SPY vs SPY II, THE GOONIES. Mam Atari 800XL. **Piotr Kowalewski, ul. Łukasińskiego 59 43-300 Bielsko Biała**

Bardzo proszę o przysłanie mi dokładnej mapy gry THE WAY OF THE EXPLODING FIST III. Szukam również gier STRIKE FORCE COBRA, THE LAST NINJA i STEEL THUNDER. W zamian wiele innych gier. Posiadam Commodore 64 i magnetofon. **Sebastian Urban, ul. Kujawska 2 m 13 84-230 Rumia-Janowo, woj. gdańskie**



FORGOTTEN WORLDS

Cały cywilizowany świat znalazł się w niebezpieczeństwie. Zły cesarz Bios (oryginalne nazwisko, czyż nie?) stworzył ośmiu bogów, którzy upodobali sobie niszczenie wszystkiego, co się tylko porusza i choć trochę przypomina człowieka. Ktoś musi ich szybko powstrzymać, zanim wszystkie wielkie miasta zamienione zostaną z tętniących życiem ludzkich siedzib w martwe kamienne pustynie zapomnianego świata — Forgotten Worlds.

Wydaje się, że pokonanie bogów i Biosa nie będzie trudne. Żyją jeszcze przecież miliony zaskoczonych, ale odważnych ludzi. Powinni w końcu zebrać wszystkie swoje siły i przezwyciężyć niebezpieczeństwo. A jednak jedyne, na co potrafili się ci ludzie zdobyć, to wysłanie jednego śmiałka i wyposażenie go w broń zdolną unicestwić całe legiony potworów.

Jak zwykle w takich przypadkach, z powodu braku ochotników całą robotę musimy wykonać sami. Jeśli chcemy wytrwać, to najlepiej będzie, gdy od samego początku gry zaczniemy strzelać do wszystkiego, co się tylko pojawi na ekranie. W miarę jak posuwamy się do przodu, pojawiają się nowe stwory, uzbrojone nawet w niekierowane rakiety. Nie będzie łatwym zadaniem pokonanie pięciu etapów tej gry. Całe szczęście, że nie jesteśmy bezbronni wobec tych stworów. My także możemy zdobyć broń.

Niektóre potwory po swojej śmierci pozostawiają nam niebieskie krążki — miejscowe pieniądze o nazwie Zennie. Naprawdę warto je pracowicie zbierać, aby potem w sklepie kupić sobie nową superstrzelbę albo puszkę coli. Gdy stać nas będzie na kupno broni, to po wyjściu ze sklepu natychmiast możemy przekonać się o jej skuteczności. Mimo że ataki są coraz silniejsze, to biedne monstra giną już na skraju ekranu, dając nam szansę na ciągłe zwiększanie naszych funduszy. W ten sposób, stale zwiększając siłę rażenia, będziemy mogli zmierzyć się z Biosem i jego pomocnikami: Bogiem Zniszczenia i Złotym Smokiem.

Gra daje okazję do niezłej strzelaniny i z pewnością zadowoli nawet wybrednych strzelców, mimo że raczej nie prezentuje żadnych graficznych rewelacji, już nie wspominając o muzyce. Za to nic w niej nie trzeba odkrywać, szukać, tylko siać spustoszenie i iść naprzód. Ciekawie rozwiązano strzelanie we wszystkich kierunkach: wciśnięcie strzału w lewo lub prawo pozwala ustawić strzelanie w żądanym kierunku, gdy strzał nie jest wciśnięty, można poruszać się po ekranie. Podobnie strzela się w grze After the War. Zatem do dzieła, ratujmy ten świat...

Firma: U.S. Gold
Komputer: Spectrum, Commodore, Amiga

GIANT

TOP SECRET
DLA KAŻDEGO GRACZA

Jeśli jesteś namiętnym, zażartym, zaciętym i stukniętym graczem, czytaj Top Secret! Ten dwumiesięcznik jest rozszerzeniem „Bajtka" dla tych, którym nie wystarczy jedna mapa miesięcznie.

Top Secret jest redagowany w całości przez naszych autorów. Wśród nich postacie takie, jak Luke, Martinez, Giant, Pegaz Ass.

Co dwa miesiące 32 strony wiadomości o grach starych i nowych. Nie zawiodą się właściciele żadnego komputera!

Jeśli masz osiągnięcia w rozpracowywaniu gier — POKE-i, mapy itp., lub jesteś autorem oryginalnej gry, skontaktuj się z nami. Adres redakcji: „Top Secret", Warszawa, ul. Wspólna 61

ZAGRAJMY W TO JESZCZE RAZ

```
H
HADES NEBULA              2279,255 L
                          6513,234
                          6514,234
                          6515,234 S
                   sys       18550
HADES 2                   6871,234
                          6872,234 L
HAPPIEST DAY             52949,234
                         52950,234
                         52951,234
                   sys       52744 EN
HARD HAT MACK            16877,173 L
                       * 8472,100 L
                       *16887,173 L
HARRISON FORD            15764,167 L
HERBY                     7191,255 L
                       *11354,234
                        11355,234
                        11356,234
HELI FLIGHT              2400,32  L
HE-MAN                   12651,234
                         12652,234
                         12653,234
                   sys       17610 L
HENRY'S HOUSE             2576,1-8 LN
                          4063,173 L
                   sys       2560
H.E.R.O.                 13865,0   L
                       *19310,0    L
                         14652,25  W
HEXENKOCHE 2-            40315,221
 -CAULDRON 2             40316,248
                   sys       32777
HEXPERT                  21875,173
                         21872,173 L
HIGH NOON                18033,255 L
HIGHWAY ENCOUNTER         6690,234
                          6691,234
                          6692,234
                   sys       4103
HOUSE OF USHER            7870,60  L
                       * 6721,238  L
HOVVER BOVVER            38680,96  L
HOW TO BE A...           65356,195
                         34582,62  E
HUNCHBACK                 9521,234
                          9522,234
                          9523,234 E
                       * 9521,44   E
                       * 5704,50   E
                       * 5704,138
                       * 7870,60
                       * 2704,173
                       *22521,44
HUNCHBACK 2              21748,234
                         21749,234
                         21750,234 S
HUNTER PATROL             7282,166 S
                       * 7282,197 S

I

I-ALIEN                   5948,205 L
                          5829,205
                          5875,205 E
                          6369,205
                          6402,205
                   sys       4032
I-BALL                 *20669,169
                       *20669,238
                   sys      16939
                       *20669,234
                        20670,234
                   sys      49741 L
I.C.U.P.S.               3516,9    L
                       * 3214,234
                         3215,169
                         3216,0
                        45826,234
                        45827,169
                        45828,0
                    sys      2080 W
```

AY & GRY

Dlaczego są programy, które pomimo tego, że posiadają moduł muzyczny dla AY-greka, nie uaktywniają żadnych dźwięków w trakcie swego działania?

Dzieje się tak dlatego, ponieważ **AY** w formie przystawki muzycznej czyli interface'u jest wytworem polskich majsterkowiczów. Oryginalnie ten chip muzyczny montowany jest w Spectrum 128 (+2, +3) i stąd mnogość melodii i efektów dźwiękowych w kodach wielu programów. Niestety zachodni producenci gier nie wiedzą o istnieniu przystawek typu **SOUND 128**, dlatego muzyka uruchamia się tylko na Spectrum posiadającym 128KB pamięci.

Programy, które posiadają pełną gamę efektów dźwiękowych przeważnie są oddzielnymi wersjami tylko dla 128K, lecz nie jest to regułą. Zdarzają się gry, które mimo, że sprawdzają na jakim typie komputera zostały uruchomione nie wykorzystują dodatkowej pamięci jaką oferuje Spectrum 128. Wtedy wystarczy tylko przeanalizować procedurę testującą typ maszyny i odpowiednio ją modyfikując można „oszukać" komputer. Tak przerobiona procedura zawsze wykaże, że program ma do czynienia z rozbudowaną wersją „trumny". Bardzo ważnym jest to, aby komputer nie wykorzystywał banków pamięci jakimi dysponuje stodwudziestkaósemka, ponieważ w „zwykłym" Spectrum tych banków nie ma i gra na pewno zawiesi się.

Oto kilka przykładów gier, które mogą używać **AY**-greka na Spectrum 48K.

Wspaniale opracowana gra **EXOLON**, mimo iż bez żadnych przeróbek posiada efekty dźwiękowe, nie zachowuje się tak, jak na Spectrum 128. Aby pomóc jej w tym, należy do komórki o adresie **28005** wpisać wartość **109**. Otrzymamy wtedy pełną — dźwiękową wersję.

Gra tego samego autora, co poprzednia, o nazwie **CYBERNOID** (część pierwsza) nie wydaje z siebie żadnych odgłosów. Zmienić to możemy przez **POKE 28030,109**, który wpisujemy przed ostatnim RANDOMIZE w loaderze gry.

VICTORY ROAD przywołujemy do wydawania dźwięków przez **POKE 52106,201**. Ta gra nie posiada żadnej melodii, lecz realistyczne odgłosy walki.

Ostatnią dziś omawianą grą jest **GARY LINEKER'S HOT SHOTS**. Ma ona bardzo ładną muzykę, którą uruchomimy przez **POKE 48534,0** i wykonaniu **RANDOMIZE USR 48510**. Te dwie instrukcje wpisujemy też przed ostatnim RANDOMIZE w loaderze.

Smutnym jest to, że mało jest gier „chodzących" tylko na Spectrum 48 z dołączonym **AY**-grekiem. Czy należy się tym martwić? Na pewno tak, ponieważ melodie jakie piszą panowie Whittaker lub Benn Daglish są przyjemne dla ucha, a efekty wzbogacają gry.

Może problem ten zniknie dzięki programowi **SOUND TRACKER** dostępnemu w naszej redakcji, a umożliwiającym tworzenie i kompilowanie muzyki pod dowolny adres w pamięci.

BRØMBA

SPECTRUMOWA LISTA PRZEBOJÓW

To przedostatnia (w tym roku) SLP. Kto wie, czy nie przedostatnia w ogóle. Coraz więcej jest głosów krytycznych, niż pochwalnych. Spotkamy się za miesiąc i podejmiemy decyzję.

Grafika:

1. SAVAGE
2. PURPLE SATURN DAY
3. CAPITAIN BLOOD
4. STRIDER
5. HARD DRIVIN

Muzyka 48:

1. CHASE H.Q.
2. SAVAGE
3. TOP GUN
4. ZANTHRAX
5. FIREFLY

Muzyka 128:

1. F.I.R.E.
2. UNTOUCHABLES
3. GLIDER RIDER
4. PLATOON
5. BATMAN THE MOVIE

Użytki:

1. SOUND TRACKER
2. ART STUDIO
3. TURBO PASCAL 3.0
4. ZEUS ASSEMBLER
5. SUMMER MONITOR

DEKODER TRANSMISJI RADIOWEJ

Na życzenie Czytelników, Słuchaczy, Wielkiego Szu i nie tylko, po raz drugi drukujemy program Dekodera Transmisji Radiowej.

Został on opracowany przez firmę SZOK ze Świebodzina specjalnie dla potrzeb słuchaczy Radiokomputera.

Aby uzyskać działający DTR+, należy:

1. Wpisać krótszy listing (Pilot) i nagrać go **SAVE *"DTR+"LINE 1**
2. Wpisać dłuższy listing i po uruchomieniu włączyć nagrywanie w celu zarejestrowania pliku kodu.

To wszystko. Teraz nagrane z radia programy należy odkodować ładując je na DTR-a i nagrywając w zwykłym formacie. Pewność wgrywania jest bardzo duża.

```
 100 CLEAR 64000: LOAD "DTR+"CODE
 110 CLS : PRINT ,,"   DEKODER TRANSMISJ
I RADIOWEJ",,,"  L-LOAD",,"  S-SAVE",,"
 BREAK-STOP",,"  R-RESET"
 120 RANDOMIZE USR 64940
```

Listing 1 - Pilot

```
 150 CLEAR 64000: LET s=0: FOR a=64940 T
O 65534: READ b: POKE a,b: LET s=s+b: NE
XT a
 160 IF s=69549 THEN  SAVE "DTR+"CODE 64
940,595: STOP
 170 PRINT "Popraw dane": STOP
 200 DATA 1,16,13,205,10,32,17,196,253,2
05,55,254,62,16,50,71,80,62,8,50,39,87,2
4,48,62
 210 DATA 83,90,79,75,60,13,13,32,32,32,
32,32,32,32,83,119,105,101,98,111,100,12
2,105,110,32
 220 DATA 32,49,57,56,56,114,174,194,9,2
04,9,219,9,237,9,0,0,0,0,0.0,0,253,54,56

 230 DATA 0,253,203,48,222,49,229,253,62
,56,50,130,88,50,98,88,205,91,254,245,20
5,65,254,241,254
 240 DATA 82,202,0,0,254,76,40,78,254,83
,204,156,255,24,221,1,21,0,205,10,32,24,
13,203,39
 250 DATA 60,33,228,253,35,61,32,252,94,
35,86,62,32,215,26,19,203,127,40,249,203
,191,215,201,1
 260 DATA 21,0,17,32,0,24,6,1,8,0,17,159
,1,205,10,32,62,32,215,27,122,179,32,248
,201
 270 DATA 33,8,92,54,0,126,254,0,40,251,
201,205,73,254,253,54,56,0,62,248,50,98,
88,243,17
 280 DATA 5,0,33,239,253,14,101,205,231,
254,33,68,95,237,91,241,253,25,218,144,2
55,33,248,92,14
 290 DATA 86,205,231,254,1,8,0,205,10,32
,253,54,57,13,33,248,92,34,237,253,229,2
37,91,239,253
 300 DATA 167,237,82,40,49,253,53,57,202
,144,255,225,229,126,254,4,210,144,255,2
05,39,254,225,6,10
 310 DATA 35,126,254,32,56,4,254,128,56,
2,62,32,215,16,241,62,13,215,35,94,35,86
,1,5,0
 320 DATA 9,25,24,194,253,53,56,251,62,7
,211,254,195,252,253,217,49,227,253,22,1
1,46,50,62,25
 330 DATA 38,1,205,76,255,48,243,45,32,2
44,62,22,22,13,205,93,255,217,185,217,32
,228,22,14,62
 340 DATA 19,24,2,62,17,205,93,255,217,1
19,169,79,35,27,122,179,217,32,240,217,1
75,185,32,110,201
 350 DATA 71,16,254,203,18,63,203,26,62,
8,203,122,32,1,122,211,254,62,127,219,25
4,31,48,94,1
 360 DATA 64,37,219,254,161,187,32,4,16,
248,24,67,95,201,38,3,205,35,255,62,28,3
7,32,248,96
 370 DATA 205,35,255,120,132,254,44,201,
205,74,255,48,41,62,25,205,84,255,56,249
,62,25,205,84,255
 380 DATA 111,62,26,205,84,255,189,38,3,
48,1,37,62,24,46,128,205,76,255,203,29,6
2,25,38,3
 390 DATA 48,245,125,201,203,66,194,232,
254,205,73,254,17,37,21,205,31,254,195,2
23,254,253,203,56,126
 400 DATA 202,252,253,237,115,61,92,62,2
48,50,130,88,17,162,9,205,31,254,205,91,
254,205,65,254,33
 410 DATA 248,92,34,237,253,237,91,239,2
53,167,237,82,202,252,253,221,42,237,253
,205,246,255,17,17,0
 420 DATA 62,0,205,194,4,221,42,237,253,
205,246,255,221,94,11,221,86,12,213,1,17
,0,221,9,221
 430 DATA 229,62,255,205,194,4,225,209,2
5,24,197,17,255,255,122,179,200,27,24,25
0
```

Listing 2 - Kod maszynowy

OPERACJE DYSKOWE W SYSTEMIE CP/M PLUS

cz. V — odczyt dyskietek MS DOS-u

W części piątej zajmiemy się organizacją dyskietki systemu MS DOS funkcjonującego na komputerach klasy IBM PC. Informacje podane tutaj będą istotne nie tylko dla właścicieli komputerów Amstrad. Posługując się danymi przedstawionymi w poprzednich częściach artykułu (Bajtek 1—8/90) podamy i opiszemy bibliotekę procedur pomocnych przy obsłudze dyskietki MS DOS'u. Efektem finalnym będzie krótki program demonstrujący możliwości pakietu.

ORGANIZACJA DYSKIETKI MS DOS'u

Typowa dyskietka używana w komputerach IBM PC ma rozmiar 5.25" cala, 40 ścieżek i 9 sektorów (512B) na ścieżkę. Numery logiczne sektorów zaczynają się od 1. Fizycznie odpowiada to dokładnie formatowi PCW 40T opisywanemu w części pierwszej (Bajtek 1—2/90). Dyskietki jednostronne (1S) odpowiadają formatowi PCW System, a dyskietki 3.5", 80 ścieżek — PCW 80T.

Instalując odpowiedni format przy pomocy programu SetFormat jesteśmy w stanie odczytywać sektory dyskietek IBM PC na CP/M'owych komputerach firmy Amstrad. Może do tego celu służyć np. program Knife Plus opisywany w Bajtku 9/89. Należy go tylko „oszukać" na początku pracy, zaczynając jego uruchomienie od typowej dyskietki Amstrada. Później można zmienić dyskietkę na IBM'ową i oglądać, kopiować poszczególne sektory.

Niestety logiczna organizacja dyskietki IMB'a jest zupełnie inna niż w CP/M'ie i dlatego nie jest możliwe kopiowanie całych plików z systemu MS DOS na Amstradach.

Cała przestrzeń na dyskietce MS DOS'u jest podzielona na 4 części (rys. 1). Mamy najczęściej jeden zarezerwowany sektor inicjujący (Boot Sector), dwie kopie tablicy FAT (File Allocation Table), 7 sektorów katalogu. Pozostałe sektory przeznaczone są na dane.

SEKTOR ZEROWY (BOOTSECTOR)

Istotne fragmenty zerowego sektora inicjującego zostały przedstawione w TAB. 1. Zebrane tam informacje pozwalają na ustalenie wszystkich parametrów koniecznych do odczytu dyskietki, a w szczególności liczby bajtów na sektor, sektorów na ścieżkę, pozycji katalogu i wielu innych. Dodatkowe informacje zakodowane są w bajcie o nazwie Media Flag, którego opis zebrano w TAB. 2.

TABLICA ALOKACJI (FAT)

Katalog dyskietki IBM nie zawiera pełnych informacji o położeniu danego pliku. W każdej pozycji katalogu (zostanie on opisany dalej) znajdują się jedynie dane o alokacji pierwszego bloku danego zbioru. Dalsze informacje znajdują się w tablicy FAT (File Allocation Table). Jest ona zapisywana na kolejnych sektorach dyskietki (rys. 2).

Na rysunku 2a przedstawiono numerację i opis sektorów typowej dyskietki (360KB, 2S, 40T, 9S/T). Z kolei rys. 2b zawiera takie dane dla dyskietki jednostronnej. Jak widać w obu przypadkach tablica FAT przechowywana jest dwukrotnie (oryginał i kopia). Dane w tej tablicy są zakodowane, na każdy blok przeznaczono 12 bitów. Dlatego chcąc uzyskać „czytelną" tablicę FAT musimy dokonać jej dekompresji, posługując się metodą zilustrowaną na rys. 3.

Z każdych trzech kolejnych bajtów tej tablicy tworzymy dwa elementy (2-bajtowe liczby całkowite) opisujące następny blok alokacji (ang. Cluster) danego pliku. Zajęte bloki numerowane są liczbami 2-4080 (dla dyskietek). Złe bloki zaznaczane są liczbami z zakresu FF1-FF7. Z kolei ostatnie bloki danego pliku mają wpisane liczby FF8-FFF, a bloki wolne opisywane są przez 000. Dodatkowe informacje dotyczące tablicy FAT można znaleźć w pliku IBM01.SYS (Listing 1).

KATALOG

Katalog dyskietki przechowywany jest w sektorach następnych w stosunku do tych, w których znajdują się tablice FAT. Dla dyskietki dwustronnej jest to 7 sektorów, a dla jednostronnej — 4 sektory. Każdy opis pliku zajmuje podobnie jak w CP/M'ie 32 bajty, ale ich znaczenie jest odmienne.

Pozycja katalogu opisana jest na rys. 4a i 4b. Pierwszy z nich stanowi mapę kolejnych bajtów, a drugi opisuje poszczególne pola. Nie wykorzystano 10 bajtów każdej pozycji katalogu, rezerwując je dla późniejszych wersji systemu operacyjnego. Podobnie jak w systemie CP/M na nazwę pliku przeznaczono 11 bajtów (8 — nazwa, 3 — rozszerzenie), zrezygnowano jednak z bajtu użytkownika (ang. User). Nie podano także bloków alokacji, wprowadzając tylko informację o pierwszym bloku zarezerwowanym przez plik. Dodatkowo w opisie pliku znalazły się bajty atrybutu opisane na rys. 5. Dzięki temu możliwa jest drzewiasta struktura katalogu znacznie bardziej elastyczna od sztywnej struktury użytkowników w CP/M'ie.

Cztery bajty katalogu przeznaczone na długość pliku pozwalają na jej przechowywanie z dokładnością do jednego bajtu. Znakowanie czasowe plików MS DOS'u możliwe jest dzięki bajtom T0, T1 i D0, D1, które zostały opisane na rys. 6a i 6b. Podano także wzory pozwalające na wyliczanie tych bajtów dla dowolnej daty i czasu.

PLIK IBM01.SYS

Odczyt pliku możliwy jest po analizie zarówno katalogu, jak i tablicy FAT. Aby ułatwić to zadanie na listingu 1 przedstawiono bibliotekę procedur IBM01.SYS. Zgrupowano w niej 15 pomocniczych procedur pozwalających na obsługę dyskietki MS DOS'u we własnych programach. Podane deklaracje stałych, typów i zmiennych pascalowych umożliwiają pełniejsze zrozumienie prezentowanych zagadnień. Kluczową rolę odgrywają procedury ReadDOSSector i ReadCluster. Obie odwołują się do procedury niskiego poziomu DD_Read. Zastąpienie tej ostatniej przez procedurę charakterystyczną dla danego komputera pozwala na uruchomienie na nim programów obsługi dyskietki MS DOS'u. Wbrew pozorom jest to nawet interesujące na IBM PC, jeśli chcemy naśladować pana Petera Nortona lub pisać nowe programy typu XTREE, Norton Commander, Norton Utilities i inne.

Ze względu na duże rozmiary artykułu, a szczególnie listingów, nie podano procedur zapisu. Nagłówki brakujących funkcji i procedur znajdują się w TAB. 3. Możliwe jest napisanie wspomnianych procedur w oparciu o podane informacje.

PLIK IBMDEMO.PAS

Krótką, ale praktyczną ilustracją omawianych zagadnień i możliwości pakietu IBM01.SYS jest program demonstracyjny IBMDemo przedstawiony na listingu 2. Pozwala on na odczyt dyskietek jedno- i dwustronnych. Najpierw wyświetlane są informacje zawarte w sektorze inicjującym. Po wyświetleniu katalogu i tablicy FAT, program pozwala na kopiowanie z dyskietek MS-DOS'a dowolnego bloku (Cluster). Tak przeniesione bloki można połączyć w plik na dysku CP/M'u posługując się danymi zawartymi w wyświetlonej tablicy FAT i katalogu.

PODSUMOWANIE

W trakcie pisania tych artykułów popełniono pewne błędy. Na ich część zwrócili uwagę niektórzy Czytelnicy. Najbardziej chyba przykrym błędem jest informacja o tym, że tablice DPB dla wszystkich komputerów firmy Amstrad są pod tym samym adresem. Tak nie jest. W szczególności dla CPC 6128 adresy te są następujące: FF64 i FF7F. Przy korzystaniu z CPC należy je wprowadzić do programu SetFormat i pliku DISK3.SYS. Jeśli programy mają chodzić na wszystkich komputerach firmy Amstrad, to należy posługiwać się funkcją BDOS o numerze 31 (adres DPB) i ewentualnie funkcją o numerze 25 (Current Drive) oraz funkcjami PCW i FDD. Te dwie ostatnie zastosowano w programie SetFormat.

W prezentowanym cyklu artykułów nie przedstawiono funkcji moduły BDOS niereализowanych przez kompilatory języków wysokiego poziomu np. Turbo Pascala. Dotyczy to m.in. wspomnianych funkcji o numerach 25 i 31, a także wielu innych.

„Serialowi" o operacjach dyskowych przyświecała idea przenaszalności zbiorów między różnymi komputerami. Z tego powodu przedstawiono informacje o ZX Spectrum (Timex FDD 3000), Spectravideo SVI-738 i IBM PC. Nie zamieszczono pełnych, „komercyjnych" programów pozwalających na transfer plików między wszystkimi omawianymi komputerami, ale prezentowane dane ułatwią ich napisanie.

Część z nich została już napisana w kręgach osób związanych z redakcją. Mam nadzieję, że będą one przedstawione na łamach poszczególnych klanów.

Jonasz Mayer

TAB. 1. Opis sektora zerowego dyskietki MS DOS'u

Bajty	Nazwa	Typowa wartość	Znaczenie
00-02	Near Jump	EB 2E 90	3 bajty skoku
03-0A	FirmName	IBM 3.1	wersja sys. op.
0B-0C	Bytes per Sector	00 02	licz. bajtów/sekt.
0D	Sectors per Cluster	02	licz. sekt/blok
0E-0F	Reserved Sectors	01 00	licz. zarezer. sekt.
10	No of FATs	02	licz. kopii FAT'u
11-12	Dir. Entries	70 00	licz. poz. katalogu
13-14	Sectors on Disk	D0 02	licz. sekt/dysk
15	Media Flag	FD	(opisany dalej)
16-17	Sectors per FAT	02 00	licz. sekt/FAT
18-19	Sectors per track	09 00	licz. sekt/ścieżkę
1A-1B	Sides	02 00	licz. stron (głowic)
1C-1D	Bad Sectors	00 00	licz. uszk. sekt.

TAB. 2. Opis bajtu MEDIA FLAG

Bit	Znaczenie
7	=1 (40 ścieżek), =0 (80)
6	=1
5	=1
4	=1
3	=1
2	=1 (dysk wymienny)
1	=1 (8 sekt/ścieżkę), =0 (9)
0	=1 (2 strony), =0 (1 strona)

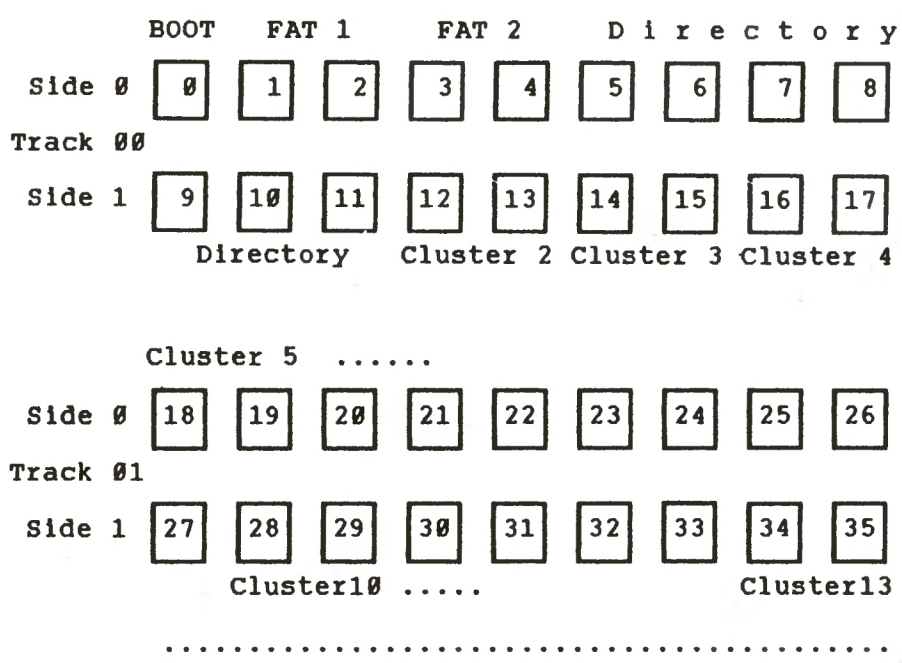
Tab.3. Lista procedur i funkcji potrzebnych do zapisu informacji na dyskietce MS DOS'u:

```
1. p. WriteDOSSector (var B : sectorType; Drive : char;
                      N : integer);
2. p. WriteCluster   (var C : ClusterType; Drive : char;
                      N : integer);
3. F. SetFTime (H,M,S : byte) : integer;
4. F. SetFDate (D,M,Y : byte) : integer;
5. p. CompressFAT (var FAT : FATType; Var eFAT : eFATType);
6. p. WriteFAT (Drive : char);
7. p. WriteDIR (Drive : char);
```

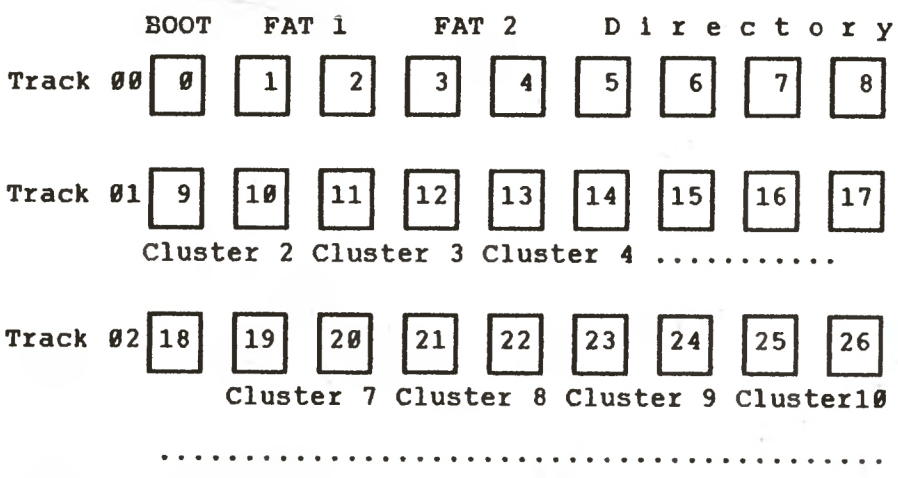
```
Sektor zerowy     (0)
Sektory FAT'u     (1,2, 3,4)
Sektory katalogu  (5,6..11)
Sektory danych    (12,13...719)
```

Rys. 1. Sektory dyskietki MS-DOS'u.
W nawiasach podano numeracje sektorów dla typowego formatu.

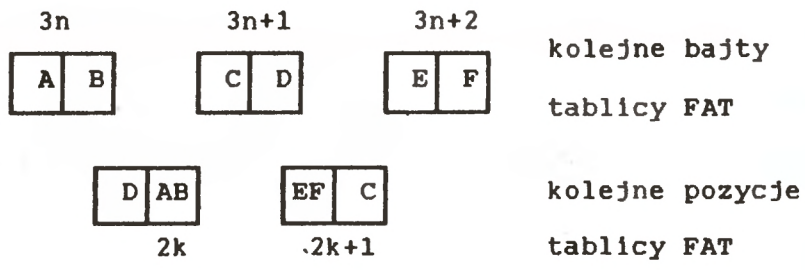
(A)



(B)



Rys. 2. Numeracja i opis sektorów dyskietki.
(A) 2-stronna, 40 ścieżek, 9 sekt/ścieżkę
(B) 1-stronna, 40 ścieżek, 9 sekt/ścieżkę



Rys. 3. Wyznaczanie dwóch kolejnych pozycji tablicy FAT na podstawie trzech kolejnych bajtów tej tablicy.

(A)

```
    0 ..... 7   8 .. 10  11   12 ...  15
    N N N N N N N N   E  E  E   A   X  X  X  X

   16........21   22 23  24 25  26 27  28 29 30 31
    X X X X  X X   T0 T1  D0 D1  F0 F1  S0 S1 S2 S3
```

(B)

```
N - Name        0-7   (nazwa            )
E - Extension   8-10  (typ              )
A - Attribute    11   (atrybut          )
T - Time       22-23  (czas zapisu      )
D - Date       24-25  (data zapisu      )
F - First      26-27  (pierwszy kluster)
S - Size       28-31  (rozmiar          )
X - not used          (nie używane      )
```

Rys. 4. Pozycja katalogu (ang. directory entry) opisująca plik dla dyskietki MS-DOS'u.

(A) Mapa kolejnych bajtów opisujących plik,
(B) Znaczenie poszczególnych pól.

```
01 - Read Only (tylko do odczytu)
02 - Hidden    (ukryty)
04 - System    (systemowy)
08 - Label     (etykieta dysku)
10 - SubDir    (podkatalog)
20 - arch.     (archiwalny)
```

Rys. 5. Opis bajtu atrybutu dla pliku.

(A)

```
      starszy bajt        młodszy bajt

   H H H H H  M M M   M M M  S S S S S

  Time := (H shl 9) + ((M shl 5) and $7E0)
```

(B)

```
      starszy bajt        młodszy bajt

   Y Y Y Y Y Y  M M   M M M  D D D D D

  Date := (Y shl 9) + ((M shl 5) and $1E0)
        + (D and $1F)
```

Rys. 6. Kodowanie czasu (A) i daty (B) dla plików w systemie MS DOS. Wzory podano w "konwencji pascalowej". Dla (A): H - godziny (hours), M - minuty (minutes), S - sekundy (seconds). Dla (B): Y - lata (years), M - miesiące (months), D - dni (days). Lata są liczone od roku 1980.

LISTING 1 Plik IBM01.SYS

```
{*****************************************************************}
{                                                                 }
{       Plik IBM01.SYS    (C) JM 1990                             }
{                                                                 }
{ Zawiera procedury umozliwiajace odczyt dyskietki MS DOS'u.      }
{                                                                 }
{ Lista procedur:                                                 }
{  1. Procedure DefineIBM (var DPB : DPBrec);                     }
{  2. procedure ReadBootSector (drive : char);                    }
{  3. procedure DisplayBootSector;                                }
{  4. procedure ReadDOSSector (var B : SectorType;                }
{                              drive : char; N : integer);        }
{  5. procedure ReadCluster (var C : ClusterType;                 }
{                            drive : char; ClusterNo : integer);  }
{  6. Function FD (E : DirEntry) : Str12;                         }
{  7. procedure GetFtime (Var H,M,S : byte; time : integer);      }
{  8. procedure GetFDate (Var D,M,Y : byte; date : integer);      }
{  9. Function Attrib (b : integer) : Str12;                      }
{ 10. Function Fsize (s,t : integer) : real;                      }
{ 11. procedure DecompressFAT (var EFAT : EFATtype;               }
{                              Var  FAT : FATtype);               }
{ 12. Procedure ReadFAT (drive : char);                           }
{ 13. Procedure ReadDIR (drive : char);                           }
{ 14. procedure DisplayDir;                                       }
{ 15. procedure DisplayFat;                                       }
{                                                                 }
{  Uwaga,                                                         }
{ Wymagany plik dodatkowy: DISK3.SYS (opis Bajtek 3-4/90).        }
{                                                                 }
{*****************************************************************}
const
     SecSize    = 511;
     CluSize    = 1023;
     MaxDirEn   = 127;
     MaxCluNo   = 359;
     MaxFatSize = 1023;
     MaxFatSec  = 3;
     MaxDirSec  = 7;
type
     aStr3 = array [1..3] of char;
     aStr8 = array [1..8] of char;
     Str12 = String [12];

     SectorType  = array [0..SecSize] of byte;
     ClusterType = array [0..CluSize] of byte;
     FATType     = array [0..MaxFatSize] of byte;
     eFATType    = array [0..MaxCluNo] of integer;
     BootRecord = record
                    jump            : array [0..2] of byte;
                    Firm            : aStr8;
```

```
                    BytesPerSector   : integer;
                    SectorPerCluster : byte;
                    ReservedSectors  : integer;
                    NumFATs          : byte;
                    NumDir           : integer;
                    SectorsOnDisk    : integer;
                    MediaFlag        : byte;
                    SectorsPerFAT    : integer;
                    SectorsPerTrack  : integer;
                    SidesOfDisk      : integer;
                    BadSectors       : integer;
                   end;
     DirEntry   = record
                    fname       : aStr8;
                    ext         : aStr3;
                    attr        : byte;
                    NotUsed     : array [0..9] of byte;
                    ftime       : integer;
                    fdate       : integer;
                    first       : integer;
                    size0,size1 : integer;
                   end;
     DirType    = array [0..MaxDirEn] of DirEntry;
var
     BootSector : SectorType;
     FatSectors : array [ 1..MaxFatSec] of SectorType;
     DirSectors : array [ 1..MaxDirSec] of SectorType;

     B : ^BootRecord;
     F : ^FATType;
     D : ^DirType;

     EFAT : eFATType;

                    { Zmienne inicj. w proc. ReadBootSector }
     SecPTrk,
     DSec,               { liczba sektorow katalogu }
     FSec,               { liczba sektorow FAT'u    }
     SecPClu,            { liczba sekt. na Cluster  }
     qFat,               { liczba kopii FAT'u       }
     NoOfClusters,       { liczba Clusterow         }

     fFatSector,         { pierwszy sektor FAT'u    }
     fDirSector,         { pierwszy sektor katalogu }
     fDataSector,        { pierwszy sektor danych   }

     FreeClusters   : integer;

Procedure DefineIBM (var DPB : DPBrec);
{*********************************************************}
{ Definiuje tablice DPB opisana w ponizszych komentarzach.  }
{*********************************************************}
begin
  with DPB    { *** IBM format compatible for drive B;   *** }
  do begin
       LSPT := $24;              { 36 sektorow log. na sciezke }
       BSH  := $04;  BLM := $0F;         { Blok alokacji 2KB }
       EXM  := $01;                       { 32 KB na segment }
       DSM  := 176;                     { 177 blokow (354 KB) }
       DRM  := $7F;                    { 128 pozycji katalogu }
       ALO  := $C0; { Dwa pierwsze bloki alokacji na katalog }
       CKS  := $20;       { Wielkosc sumy kontrolnej katalogu }
       OFF  := $01;                  { 1 zarezerwowana sciezka }
       PSH  := $02;  PHM := $03;     { sektor fiz. 512 bajtow }
       Sideness := $81;          { Dwustronny naped flip flop }
       TPS  := $28;                    { 40 sciezek na strone }
       PSPT := $09;       { 09 sektorow fizycznych na sciezke }
       FSN  := $01;                 { numer pierwszego sektora }
       SS   := $0200;             { sektor fizyczny 512 bajtow }
       RWGAP:= $2A;  FGAP:= $52;       { przerwy r/w i format }
       MFM  := $60;                      { zapis w trybie MFM }
       AUTO := $00;                    { autodetekcja formatu }
     end;
end; { of Define DPB }

procedure ReadBootSector (drive : char);
{*********************************************************}
{ Wczytanie sektora zerowego dyskietki MS DOS'u.          }
{ Inicjalizacja istotnych zmiennych.                      }
{*********************************************************}
begin
  DD_Read (drive,0,0);               { read boot sector }
  Move (Buf,BootSector,512);
  B := PTR (addr (BootSector));
  with B^
  do begin
       SecPTrk     := SectorsPerTrack;
       DSec        := NumDir div 16;   { Sector of 512 b only }
       FSec        := SectorsPerFAT;
       SecPClu     := SectorPerCluster;
       qFat        := NumFATs;

       fFatSector  := ReservedSectors;          {  1 }
       fDirSector  := fFatSector + qFAT * FSec; {  5 }
       fDataSector := FDirSector + DSec;        { 12 }
       NoOfClusters := (SectorsOnDisk - fDataSector) div
                            SectorPerCluster;
     end;
end; { Read Boot Sector }

procedure DisplayBootSector;
{*********************************************************}
{ Wyswietlenie sektora zerowego dyskietki MS DOS'u.       }
{*********************************************************}
begin
  with B^
  do begin
       Writeln ('FIRM;             ',Firm:8);
       writeln ('Bytes per Sector;    ',BytesPerSector:4);
       writeln ('Sectors per Cluster; ',SectorPerCluster:4);
       writeln ('Reserved Sectors;    ',ReservedSectors:4);
       writeln ('No Of FATs;          ',NumFATs:4);
       writeln ('Sectors Per FAT;     ',SectorsPerFAT:4);
       writeln ('Dir Entries;         ',NumDir:4);
       writeln ('Sectors on Disc;     ',SectorsOnDisk:4);
       writeln ('Sectors per Track;   ',SectorsPerTrack :4);
       writeln ('Media Flag;          ',(MediaFlag):4);
       writeln ('Sides;               ',SidesOfDisk:4);
       writeln ('Bad Sectors;         ',BadSectors:4 );
     end;
end; { Display Boot Sector }

procedure ReadDOSSector (var B : SectorType; drive : char;
                             N : integer);
{*********************************************************}
{ Odczyt sektora DOS'owego dyskietki MS DOS'u.            }
{*********************************************************}
var
   track, sector : byte;
begin
       track  := N div SecPTrk;
       sector := N mod SecPTrk;;
       DD_Read ( drive, track, sector);
       Move (Buf, B, 512);
end; (* read DOS sector *)

procedure ReadCluster (var C : ClusterType; drive : char;
                           ClusterNo : integer);
{*********************************************************}
{ Odczyt zadanego clustera z dyskietki MS DOS'u.          }
{*********************************************************}
var i, DSect : integer;
    track, sector : byte;
begin
  DSect := (ClusterNo-2)*SecPClu + fDataSector;
  for i := 1 to SecPClu
  do begin
       track  := DSect div SecPTrk;
       sector := DSect mod SecPTrk;
       DD_Read (drive,track,sector);
       Move (Buf, C [(i-1)*512], 512);
       Dsect := Dsect + 1;
     end;
end; { of Read Cluster }

Function FD (E : DirEntry) : Str12;
{*********************************************************}
{ Zwraca nazwe pliku danej pozycji katalogu.              }
{*********************************************************}
var s : str12;
    i : integer;
begin
  with E
  do begin
  for i := 1 to 8 do s[i]   := chr (ord (E.Fname[i]) and $7F);
  s[9] := '.';
  for i := 1 to 3 do s[i+9] := chr (ord (E.Ext[i])   and $7F);
  s[0] := chr (12);
     end;
  FD := s;
end; { of F }

procedure GetFtime (Var H,M,S : byte; time : integer);
{*********************************************************}
{ Wyznaczenie czasu zapisu pliku MS DOS'u.                }
{*********************************************************}
begin
  S := (time and $1F) shl 1;     { seconds }
  M := (time shr 5) and $3F;     { minutes }
  H := time shr 11;              { hours   }
end; { GET IBM FILE TIME }

procedure GetFDate (Var D,M,Y : byte; date : integer);
{*********************************************************}
{ Wyznaczenie daty zapisu pliku MS DOS'u.                 }
{*********************************************************}
begin
  D := date and $1F;              { day   }
  M := (date shr 5) and $0F;      { month }
  Y := (date shr 9) + 80;         { year  }
end; { GET IBM FILE DATE }
```

```
243;
244; Function Attrib (b : integer) : Str12;
245; {*************************************************************}
246; { Wyznaczenie atrybutu pliku MS DOS'u.                        }
247; {*************************************************************}
248; var a : str12;
249; begin
250;   a := '';
251;   if b and $01 = $01 then a := a + 'r' else a := a + '.';
252;   if b and $20 = $20 then a := a + 'a' else a := a + '.';
253;   if b and $04 = $04 then a := a + 's' else a := a + '.';
254;   if b and $02 = $02 then a := a + 'h' else a := a + '.';
255;   if b and $10 = $10 then a := ' dir';
256;   if b and $08 = $08 then a := ' lab';
257;   Attrib := a;
258; end; { attribute }
259;
260; Function Fsize (s,t : integer) : real;
261; {*************************************************************}
262; { Wyznaczanie rozmiaru pliku MS DOS'uw bajtach                 }
263; {*************************************************************}
264; var x,y : real;
265; begin
266;   x := s; if x<0 then x := x + 65536.0;
267;   y := t; if y<0 then y := y + 65536.0;
268;   Fsize := x + 65536.0 * y;
269; end; { of File Size in bytes }
270;
271; procedure DecompressFAT (var EFAT : EFATtype; Var FAT : FATtype
272; {*************************************************************}
273; { Dekompresja tablicy FAT, 12-bitowe pozycje zamieniane sa    }
274; { na liczby calkowite (16-bitowe).                            }
275; {*************************************************************}
276; var i,k, y : integer;
277; begin
278;   i := 0;  k := 0;            FreeClusters := 0;
279;   repeat
280;     y := FAT [k+1];
281;     EFAT [i]   := FAT [k] + (y and $0F) shl 8;
282;     EFAT [i+1] := FAT [k+2] shl 4 + (y shr 4);
283;     if EFAT [i] = 0 then FreeClusters := FreeClusters + 1;
284;     if EFAT [i+1]=0 then FreeClusters := FreeClusters + 1;
285;     i := i + 2;
286;     k := k + 3;
287;   until i >= NoOfClusters + 2;
288; end; { of Decompress FAT }
289;
290; Procedure ReadFAT (drive : char);
291; {*************************************************************}
292; { Wczytanie sektorow  tablicy FAT.                             }
293; {*************************************************************}
294; var i : integer;
295; begin
296;   for i := 1 to FSec
297;   do ReadDOSsector (FATsectors [i], drive, i + fFatSector -1);
298;   F := PTR (addr (FatSectors));
299;   DecompressFAT (EFAT,F^);
300; end; { Read Directory }
301;
302; Procedure ReadDIR (drive : char);
303; {*************************************************************}
304; { Wczytanie sektorow katalogu.                                 }
305; {*************************************************************}
306; var i : integer;
307; begin
308;   for i := 1 to DSec
309;   do ReadDOSsector (DIRsectors [i], drive, i + fDirSector -1);
310;   D := PTR (addr (DIRsectors));
311; end; { Read Directory }
312;
313; procedure DisplayDir;
314; {*************************************************************}
315; { Wyswietla katalog dyskietki.                                 }
316; {*************************************************************}
317; var i, FileNo : integer;                    ch : char;
318;     hour,min,sec, day,mon,year : byte;
319;     size : real;
320;     name, atr : str12;
321; begin
322;   ClrScr;
323;   writeln ('Directory of IBM disk');
324;   writeln;
325;   FileNo := 0;
326;   for i := 0 to DSec*16-1
327;   do with d^[i]
328;      do begin
329;           if not (fname[1] in [#$00,#$E5])
330;           then begin
331;                  name := FD (d^[i]);
332;                  atr  := Attrib (attr);
333;                  FileNo := FileNo + 1;
334;                  GetFTime (hour,min,sec,ftime);
335;                  GetFDate (day,mon,year,fdate);
336;                  size := Fsize (size0,size1);
337;      writeln (name,size:10:0,day:4,'-',mon:2,'-',
338;               year:2,hour:4,':',min:2,'  ',atr,'  ',first);
339;                  if FileNo mod 28 = 0
340;                  then begin
341;                         repeat until keypressed;
342;                         read (kbd,ch);
343;                       end;
344;               end;
345;        end;
346;   writeln;     writeln (FileNo:10,' File(s). ');
347; end; { of Display Dir }
348;
349; procedure DisplayFat;
350; {*************************************************************}
351; { Wyswietla tablice FAT.                                       }
352; {*************************************************************}
353; var i : integer;
354; begin
355;   writeln ('    ',FreeClusters * SecPClu div 2,' kB free.');
356;   writeln;
357;         writeln ('FAT TABLE'); writeln;
358;         for i := 2 to NoOfClusters+1
359;         do begin
360;              if efat [i] <> 4095
361;              then write (EFAT[i]:6)
362;              else write (' <EOF>');
363;              if (i-1) mod 12 = 0 then writeln;
364;            end;
365;   if (i-1) mod 12 <> 0 then writeln;
366; end; { of Display FAT }
367;
368; {*************************************************************}
```

LISTING 2 Plik IBM DEMO,PAS

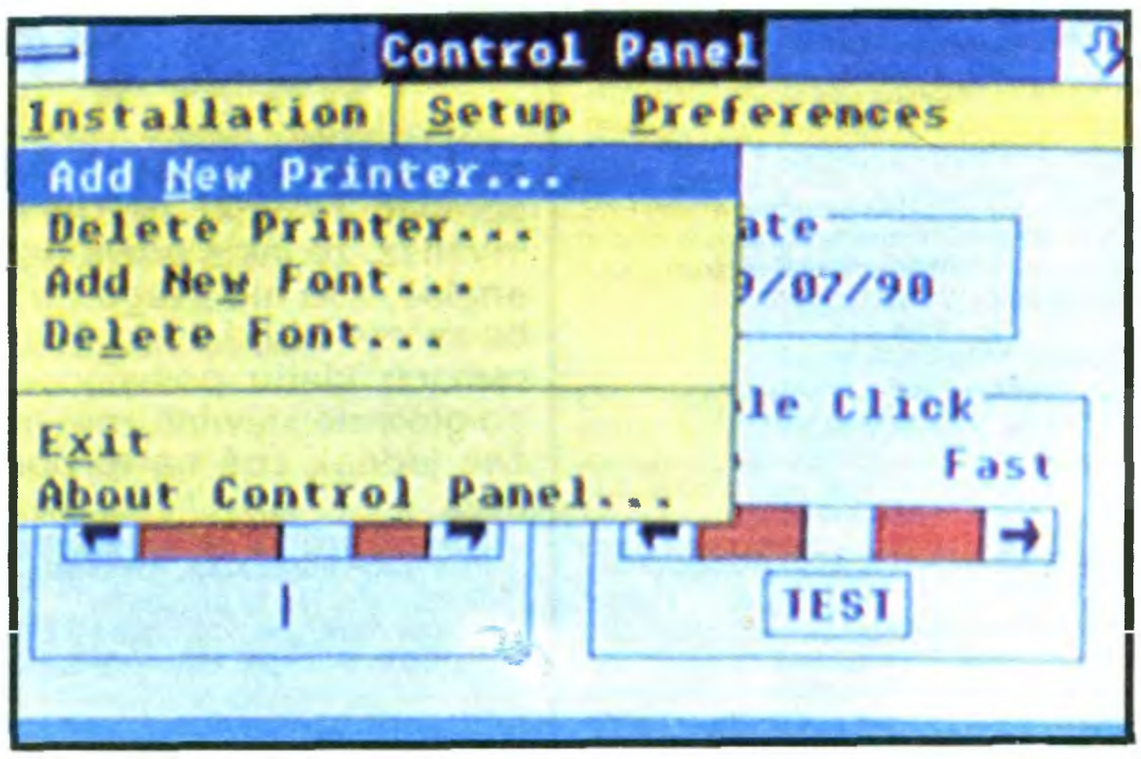
```
  1; Program IBMdemo;
  2; {*************************************************************}
  3; {                                                             }
  4; {          Plik IBMDEMO.PAS   (C) JM 1990                     }
  5; {                                                             }
  6; { Program demonstrujący możliwości biblioteki IBM01.SYS.      }
  7; {                                                             }
  8; {  Działanie:                                                 }
  9; { 1. Wybór napedu z dyskietka MS DOS'u,                       }
 10; { 2. Odczyt sektora zerowego, tablicy FAT i katalogu,         }
 11; { 3. Wyświetlenie obiektów z punktu 2,                        }
 12; { 4. Odczyt cluster'a i zapisanie go na dysku CP/M.           }
 13; {  Podanie numeru cluster'a < 2, kończy program.             }
 14; {                                                             }
 15; {  Uwaga.                                                     }
 16; { Dodatkowa biblioteka; plik DISK3.SYS.                       }
 17; {                                                             }
 18; {*************************************************************}
 19;
 20; {$I DISK3.SYS }
 21; {$I IBM01.SYS }
 22;
 23; var
 24;     drive : char;           ch        : char;
 25;     DPB   : DPBrec;         ClusterNo : integer;
 26;     Name  : String [20];    Cluster   : ClusterType;
 27;     PCWf  : file;
 28;
 29; begin
 30;   ClrScr;
 31;   write ('Enter IBM drive (A,B): ');   readln (drive);
 32;   Drive := UpCase(drive);
 33;   if not (drive in ['A','B']) then Halt;
 34;   DefineIBM (DPB);
 35;   SetDPB (drive,DPB);
 36;
 37;   ReadBootSector (drive);
 38;   ReadDir (drive);
 39;   ReadFat (drive);
 40;
 41;   DisplayBootSector;  repeat until keypressed;  read (kbd,ch);
 42;   DisplayDir;         repeat until keypressed;  read (kbd,ch);
 43;   DisplayFAT;         repeat until keypressed;  read (kbd,ch);
 44;
 45;   repeat
 46;     Write ('Cluster to read: ');
 47;     readln (ClusterNo);
 48;     if ClusterNo<2 then Halt;      { wyjscie "na butach" }
 49;
 50;     Str (ClusterNo,Name);
 51;     Name := 'Cluster.' + Name;
 52;     assign  (PCWf,Name);
 53;     Rewrite (PCWf);
 54;
 55;     ReadCluster (Cluster,Drive,ClusterNo);
 56;     BlockWrite  (PCWf,Cluster, 4*SecPClu );
 57;
 58;     Close   (PCWf);
 59;   until false;                    { ad infinitum }
 60; end.
 61;
 62; {*************************************************************}
```

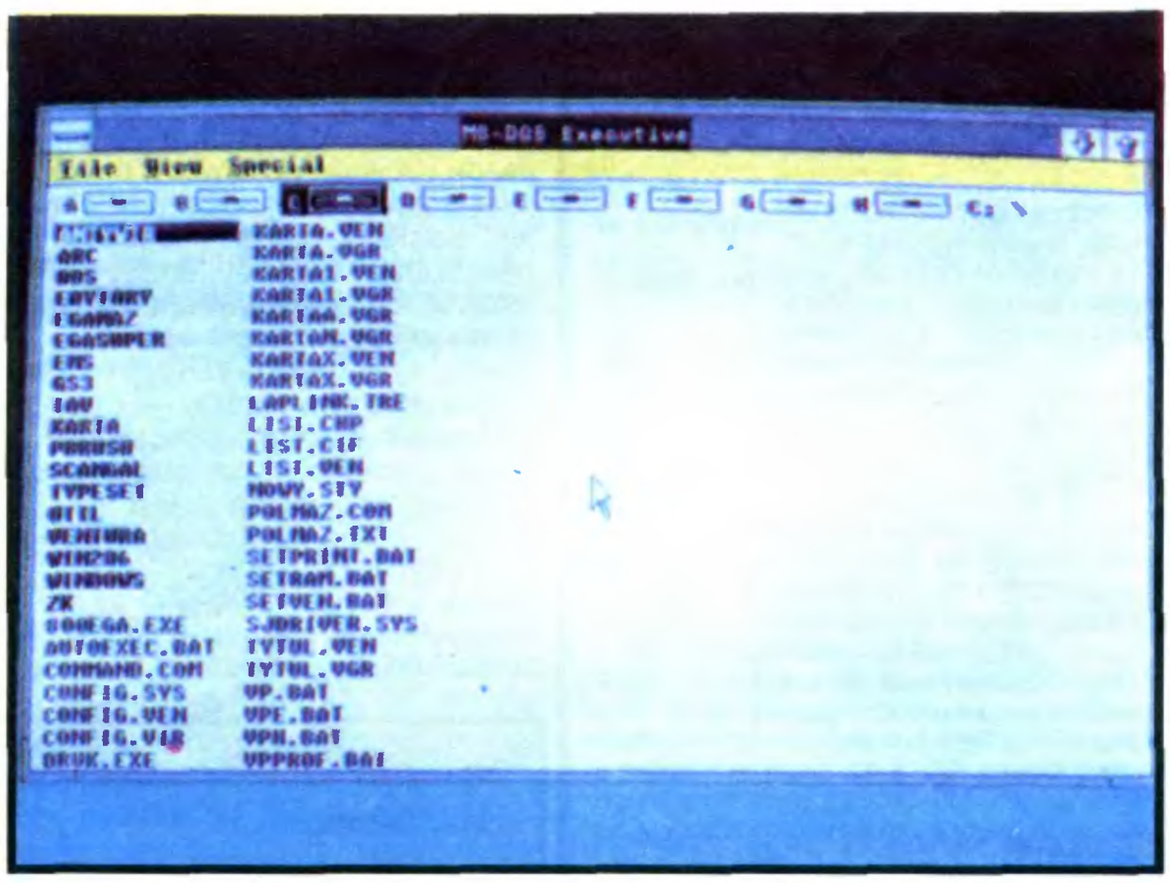
KUPIĘ, SPRZEDAM, ZAMIENIĘ

Pod tym tytułem otwieramy w Bajtku nową rubrykę. Każdy, kto przyśle do nas wycięty z Bajtka kupon (odbitek nie będziemy honorować), może zamieścić krótkie ogłoszenie. Maksymalna długość ogłoszenia — piętnaście słów razem z adresem, drobne odchylenia do zaakceptowania. Ogłoszenie może dotyczyć sprzedaży, kupna lub zamiany komputera i akcesoriów — wszelkiego typu urządzeń zewnętrznych, programów i literatury, używanych i nowych, pod warunkiem, że oferta dotyczyć będzie pojedynczych sztuk. Ogłoszenia drukować będziemy kolejno w miarę ich napływania. Zastrzegamy sobie prawo niewydrukowania oferty, w razie podejrzeń o próbę sprzedaży hurtowych ilości towaru, oraz prawo do zmiany zasad akceptowania ogłoszeń. Piszcie na nasz adres, z dopiskiem na kopercie — Kupię-Sprzedam-Zamienię.

MS WINDOWS

czyli o tym jak góra przyszła do Mahometa





Niemal każdemu z nas zdarzyło się kiedyś wgrać z dyskietki program o intrygującym tytule, entuzjastycznie opisany w jednym z zachodnich czasopism.

Jednak zamiast ekranu początkowego pojawił się enigmatyczny napis „This program requires Microsoft Windows", i program skończył pracę. Po powrocie do opisu okazuje się, że rzeczywiście petitem na dole strony jest napisane „Program must run under MS Windows". Dokładniejsza lektura czasopisma ujawnia, że przy bardzo wielu reklamach programów pojawia się podobny napis. Czym zatem są owe tajemnicze MS WINDOWS?

Wbrew pozorom nie jest to oddzielny system operacyjny, kolejny z dostępnych na IBM PC, lecz normalny program użytkowy. Najbardziej przypomina on pod względem posługiwania się bardzo rozbudowaną nakładkę na system operacyjny, coś jak NORTON COMMANDER, o specyficznych właściwościach. Mam nadzieję, że użytkownicy MS WINDOWS wybaczą mi to uproszczenie. Mam zamiar w dalszej części tekstu wykazać jakościową różnicę pomiędzy MS WINDOWS a niemal każdą inną nakładką.

PIERWSZE SPOTKANIE

Załóżmy, że kolega zostawił nam komputer (koniecznie IBM PC) razem z lakoniczną informacją, że na dysku „C": w katalogu „WINDOWS" są „okienka", a wszystkie katalogi poniżej „WINDOWS" zawierają programy, które wymagają obecności „okienek". Piszemy więc z poziomu DOS'a zlecenie WIN i wchodzimy w świat WINDOWS.

Nasz komputer powinien być wyposażony w mysz i najlepiej także w monitor kolorowy.

Po starcie programu zobaczymy copyright firmy MicroSoft, następnie komputer przejdzie w tryb graficzny i wyświetli na ekranie listę plików z aktualnego katalogu. Jeden z plików, pierwszy na liście będzie podświetlony, a na środku ekranu pojawi się graficzny kursor myszy w kształcie strzałki. Ponad listą plików będzie podany ciąg symboli stacji dysków dostępnych w naszym komputerze, a także symbol aktualnego katalogu.

Niestety, nic nie mogę powiedzieć na temat kolorów, jakie będą na ekranie, kolejności wyświetlonych plików, nawet nie wiem, jakim krojem pisma będą te nazwy podane. Jeżeli nasz hipotetyczny kolega jest miłośnikiem historii średniowiecznej, to może to być gotyk.

Załóżmy jednak, że prywatne zainteresowanie naszego kolegi nie przeszło (jeszcze) w manię i na ekranie tekst jest wypisany dającymi się odczytać czcionkami. Zatem w drugiej linii ekranu powinna pojawić się tzw. banderola, czyli poziome menu. Zawiera ona zazwyczaj tylko kilka opcji, licząc od lewej są to „FILES", „VIEW", „SPECIAL", czasami jeszcze kilka innych. Ten początkowy ekran jest oknem systemowym, tzn. realizowany jest w nim program systemu operacyjnego, przypomina nam o tym napis „MS DOS EXECUTIVE" w najwyższej linii ekranu. Wszystkie operacje systemowe są teraz banalnie proste.

Np. chcąc zmienić aktywną stację dysków najeżdżamy myszą na jej symbol (ikonę) i dwukrotnie naciskamy przycisk myszy. Chcąc zmienić katalog, postępujemy tak samo. Aby skasować plik, najeżdżamy na jego symbol myszą, podświetlamy nazwę pliku, następnie otwieramy menu „FILES" i wybieramy z niego opcję „DELETE". System będzie chciał się upewnić, poprzez wyświetlenie okienka dialogowego, czy jesteśmy zdecydowani skasować plik. Jeżeli potwierdzimy nasz zamiar poprzez „OK", to plik zostanie skasowany.

Po kilku minutach zobaczymy, że wszystkie operacje systemowe, poruszanie się po różnych dyskach, katalogach, kasowanie, kopiowanie i przemianowywanie plików nie stanowią większego problemu.

Opanowaliśmy system operacyjny w przeciągu dziesięciu minut. Bez konieczności pamiętania kolejności parametrów w komendzie „COPY", bez podawania długich list katalogów w „CD", bez liczenia na palcach ilości stacji dysków w systemie i sprawdzania, czy możemy wydać komendę „E:", bo akurat nie pamiętamy, czy założyliśmy RAM-DYSK, czy też nie.

Uruchomienie programu na tym poziomie, odbywa się poprzez podświetlenie jego nazwy i podwójne naciśnięcie myszy. Uruchomienie programu znajdującego się w innym katalogu (jeżeli nie chcemy przechodzić do tego katalogu), poprzez otwarcie menu „FILES", wybranie opcji „RUN" i wpisanie w okienko dialogowe nazwy programu, który chcemy wykonać. DOS'owa zmienna PATH jest respektowana.

Jeżeli uruchamiany program był napisany z myślą o współpracy z MS WINDOWS, to otrzyma on własne okienko, podobnie jak poprzednio system operacyjny. W tym okienku będzie prezentował wyniki, w nim też będzie prosił o wprowadzenie danych.

Jeżeli uruchamiamy zwykły program, to zazwyczaj otrzymuje on do dyspozycji cały ekran i możemy w nim pracować, jakby w pamięci nie było MS WINDOWS. Nie jest to cała prawda, w dalszej części postaram się podać algorytm pracy przy uruchamianiu programów, nie przeznaczonych do pracy z MS WINDOWS.

Wielu czytelników zdziwi się. 1 MB kodu po to, aby mieć ładniejszy ekran i łatwiej zmieniać katalogi! Nie mają racji. To, co do tej pory przedstawiłem, jest tylko zewnętrzną powłoką MS WINDOWS. Powłoka ta ustala sposób komunikacji z użytkownikiem, deklaruje standard wprowadzania i wyprowadzania danych, jest też reklamą systemu. Prawdziwe oblicze „okienek" ukryte jest głębiej.

Przejawia się ono w kilku cechach MS WINDOWS, całkowicie obcych systemowi DOS. Przede wszystkim jest to ogromna łatwość wymiany danych pomiędzy różnymi programami. Opracowanie wykresu jednym programem „BUSSINES GRAPHICS", narysowanie ładnej winiety innym programem rysunkowym, a następnie włączenie tego wszystkiego w tekst opracowywany pod edytorem nie przedstawia problemów. Oprócz tego na ekranie można otworzyć jednocześnie kilka okien i na gorąco szacować, jak będzie się miała winietka rysowana w jednym oknie do wykresu, który aktualnie powstaje w drugim. A jeżeli chcemy chwilowo przerwać rysowanie i wrócić do edytora tekstów, to nic prostszego. Wskazujemy myszą ramkę okienka edytora (tzn. wybieramy okno edytora jako aktywne) i możemy w nim pracować.

Użytkownicy pakietów zintegrowanych mogą powiedzieć, że mają to wszystko od dawna. Nie będzie to prawda. Pakiety zintegrowane są programami zamkniętymi, użytkownik może posługiwać się jedynie tymi modułami, które zostały włączone do pakietu przez projektantów. W wypadku MS WINDOWS sprawa ma się inaczej; to użytkownik wybiera programy, które chcę uruchamiać, a setki niezależnych od siebie firm software'owych opracowują kolejne programy przeznaczone do pracy pod nadzorem MS WINDOWS.

Oprócz łatwości przenoszenia danych największą zaletą MS WINDOWS jest standaryzacja sposobu komunikacji z użytkownikiem w różnych programach. Każdy program musi być sterowany w ten sam sposób. Skrajnym menu po lewej stronie ekranu MUSI być „FILES". Program może pochodzić od dowolnego producenta, ale sposób wprowadzania danych MUSI być identyczny. O wygodzie takiego rozwiązania nie trzeba nikogo przekonywać. Pod MS WINDOWS załadowanie nowego pliku odbywa się ZAWSZE poprzez wybranie ze skrajnego lewego menu „FILES" opcji „NEW".

Dobrym przykładem jest obsługa urządzeń zewnętrznych, drukarek, ploterów, skanerów itp. Jeżeli zmieniliśmy typ drukarki, to przy pracy bez MŚ WINDOWS czeka nas kilka godzin na przeinstalowanie wszystkich programów. A jeżeli nie posiadamy dyskietek instalacyjnych (ponieważ programy dostały się w nasze ręce — powiedzmy, z pewnym naruszeniem praw autorskich), co wtedy? A jeśli chcemy mieć możliwość używania dwóch różnych drukarek przy tym samym programie, to czy musimy trzymać na dysku dwie pełne kopie programu? Przy pracy bez MS WINDOWS będą same kłopoty. Jeżeli używamy „okienek", to kwestia dopisania nowej drukarki do listy zdefiniowanych w systemie jest prosta.

Po takim zabiegu wszystkie programy pracujące pod MS WINDOWS będą miały dostęp do nowej drukarki. Co najważniejsze, jest rzeczą obojętną, czy autorzy danego programu przewidzieli użycie takiej właśnie drukarki — mogła ona w momencie pisania programu wcale jeszcze nie istnieć.

Na koniec kilka danych technicznych oraz informacji dla aktualnych i przyszłych użytkowników MS WINDOWS:

— program powinien być instalowany z dyskietek instalacyjnych, polega to na serii odpowiedzi o konfigurację naszego systemu komputerowego,

— w zasadzie wymagana jest mysz, używanie MS WINDOWS bez myszy jest możliwe, ale niewygodne,

— podobnie z twardym dyskiem — bardzo ułatwia pracę,

— zalecany byłby niezły monitor kolorowy (np. EGA), chociaż MS WINDOWS potrafią obsługiwać około 100 różnych kart graficznych, a wszystkie nowe karty graficzne są sprzedawane razem z programem (driver'em), do współpracy z MS WINDOWS. W procedurze instalacyjnej jest dokładnie opisane, w jaki sposób instalować niestandardowe urządzenie w systemie,

— MS WINDOWS nie są wybredne pod względem sprzętu, dadzą się uruchomić nawet na PC XT z dwoma napędami dysków elastycznych i monitorem HERCULES, ale prawdziwy „lwi pazur" pokazują dopiero przy dużych zestawach komputerowych,

— jeżeli nasz komputer posiada pamięć rozszerzoną w standardzie EMS wersja › = 4.0, to MS WINDOWS potrafią ją wykorzystać,

— jeżeli chcemy pracować z RAM-DYSKIEM, to lepiej użyć programu RAMDRIVE.SYS, który znajduje się na dyskietkach instalacyjnych, potrafi on założyć RAM-DYSK zarówno w pamięci EXTENDED jak i EXPANDED,

— nie uruchamiajmy programów nie przeznaczonych do pracy pod MS WINDOWS po wejściu w system. W szczególności nie uruchamiajmy żadnych programów rezydentnych. Jeżeli będziemy mieli szczęście, to komputer się od razu zawiesi, jeżeli nie, to zawiesi się przy końcu opracowywania najczęściej unikalnych danych,

— niemal wszystkie programy uruchamiane pod MS WINDOWS zapisują lub odczytują plik konfiguracyjny WIN.INI. Powinien być on w katalogu macierzystym „okienek". Raczej nie należy zmieniać zawartości tego pliku ręcznie, chyba, że dokładnie wiemy, co chcemy osiągnąć i wiemy, w jaki sposób to zrobić,

— wprawdzie otwarcie naraz na ekranie kilkunastu okienek jest bardzo widowiskowe, ale tak naprawdę to nigdy nie są potrzebne więcej niż dwa — trzy, pozostałe tylko zajmują pamięć i spowalniają pracę,

— co jakiś czas wykonujemy komendę CHKDSK, czasami przy próbach uruchamiania oprogramowania nie przeznaczonego do pracy pod MS WINDOWS na dysku pozostają jakieś bezprzydziałowe sektory,

— tymczasowe pliki tworzone przez MS WINDOWS mają nazwy zaczynające się od znaku „r", po zakończeniu sesji z programem są one kasowane, ale jeżeli zakończyliśmy pracę przez restart komputera, to należy je potem wykasować samemu.

Od czego zacząć — czyli, co jest na dyskietkach systemowych?

Firma MicroSoft — producent MS WINDOWS dostarcza 10 dyskietek instalacyjnych (przynajmniej z wersją 2.03). Program instaluje się niemal samoczynnie. Wśród tych plików jest kilka programów typu EXE. Są to zarówno programy niezbędne do prawidłowego funkcjonowania systemu, jak i kilka programów użytkowych.

Najważniejsze z nich to:

— CONTROL.EXE — obsługujący MS WINDOWS od strony konfiguracji — kolory, kroje pisma, nowe drukarki itp.
— NOTEPAD.EXE — podręczny edytor tekstu, przydatny przy poprawianiu krótkich plików np. typu .SYS lub .BAT
— WRITE.EXE — odrobinę bogatszy edytor tekstu, w założeniu profesjonalny, lecz mało przydatny
— PAINT.EXE — program graficzny, mało ambitny, lecz od biedy do użytku.
— CALENDAR.EXE — kalendarz z notatnikiem
— CALC.EXE — podręczny kalkulator, do prostych przeliczeń
— TERMINAL.EXE — do nadawania i odbioru poczty elektronicznej...
— REVERSI.EXE — prosta gra logiczna, pożeracz czasu.

Wydaje mi się, że MicroSoft celowo dołączył do pakietu programy, które potencjalnie są bardzo użyteczne, ale sprawiają wrażenie nie skończonych bądź nie dopracowanych. Miałoby to na celu zachęcenie właścicieli MS WINDOWS do zakupu prawdziwie profesjonalnego oprogramowania przeznaczonego do pracy w środowisku MS WINDOWS. Te programy, to doskonałe edytory tekstu (np. MS WORD, AMI), programy graficzne (MS PAINTBRUSH, ART & LETTERS) oraz cała gama (obecnie ponad 500 pozycji) innych. Niestety, o ile same MS WINDOWS kosztują bardzo przystępnie (poniżej 100 $ za pełną instalację), o tyle np. sam MS-WORD, wersja przystosowana do pracy z „okienkami", kosztuje ponad 400 $.

Co dalej — czyli bardziej konkretne informacje techniczne, opisy kilku programów działających pod MS WINDOWS, rady dla użytkowników, programistów i zwykłych ciekawskich — w następnych numerach „Bajtka"

P.S. Skąd się wziął tytuł?

Jest grupa czytelników, którzy się ironicznie uśmiechali czytając ten tekst. Są to (niestety, bardzo nieliczni w Polsce) użytkownicy, komputera APPLE MACINTOSH.

Oni to wszystko, o czym pisałem, znają doskonale. Dla nich nie jest to jeszcze jedna nakładka na system operacyjny, ale podstawowy i naturalny tryb pracy. Znają ten tryb od 1984 roku, od powstania pierwszej wersji ich komputera. Z pewnością także pamiętają pełne ironii i kpiny opinie firm IBM i MicroSoft na temat systemu operacyjnego zainstalowanego w MACINTOSH'u i całej ideologii systemów typu „User FRIENDLY". No cóż, minęło kilka lat i obecnie pan Bill Gates — szef firmy MicroSoft publicznie oświadcza, że wszystkie nowe programy z jego firmy będą przystosowane do pracy pod MS WINDOWS, a wersja DOS'owa może już nie zawsze być dostępna. Z drugiej strony, firmie APPLE trudno dalej nazywać IBM PC pudełkiem na złom, gdzie wstawia się zakupione po różnych wyprzedażach pojedyńcze karty — MACINTOSH stał się też komputerem o otwartej architekturze. Zatem nie tyle „góra" (MicroSoft) przyszła do „Mahometa" (APPLE), ile i „góra" i „Mahomet" zaczęły podążać sobie naprzeciw.

Z.K.

STUKU PUKU

Klawiatura IBM-a jest raczej cicha i czasami może sprawić brzydki psikus — myślisz, że jakiś guzik nacisnąłeś, a tu nic z tego i trzeba zaczynać od początku. W ramach klanu poświęconego głównie klawiaturom można jednak coś na to poradzić.

Po Polsce krąży kilka, a może i kilkadziesiąt wersji programu **keyclick.com,** „stukającego" z okazji naciśnięcia jakiegoś klawisza, ale — żeby nikt się nas nie czepiał — napisaliśmy własną prymitywną wersję. Jest to krótka asemblerowa procedura, instalowana rezydentnie w systemie i ruszająca w tę i z powrotem membranę głośnika, ilekroć zostaje wywołane przerwanie 9 (patrz **Gmeranie w klawiszach).** W efekcie każde naciśnięcie klawisza powoduje dwa stuknięcia — jedno, gdy klawisz jest wciskany, drugie, gdy jest puszczany. Można by ten program napisać również w taki sposób, by reagował tylko na „prawdziwe" klawisze, i to jeden raz. Byłby wtedy jednak dwa razy dłuższy.

Program jest bardzo prosty, ale jedna instrukcja w nim może się wam wydawać podejrzana — jest nią PUSHF, przed wywołaniem BIOS-u. Jest ona niezbędna do zasymulowania przerwania, które między innymi tym się różni od zwykłego wywołania podprogramu, że przed zapamiętaniem rejestrów CS:IP i wykonaniem skoku wykonywana jest właśnie instrukcja PUSHF. Instrukcja powrotu z przerwania, IRET, powoduje (oprócz powrotu do programu wywołującego) również odtworzenie rejestru znaczników, którego w naszym przypadku nie byłoby na stosie, gdybyśmy go tam specjalnie nie umieścili.

Jak przygotować do pracy program asemblerowy, już pisaliśmy, potrzebna jest jednak tym razem pewna dodatkowa uwaga. Otóż aby membrana głośnika wydała dźwięk, musi być poruszona dwa razy — w tę i z powrotem, w pewnym odstępie czasu. Ten odstęp czasu jest realizowany za pomocą programu obsługi przerwania dziewiątego — jeśli okaże się, że program ten działa zbyt szybko, stuknięcie membrany może być prawie niesłyszalne. Trzeba wtedy wstawić za wywołaniem BIOS-u, a przed drugą instrukcją OUT niewielką pętlę opóźniającą, na przykład taką:

```
        mov cx,50h
label:
        loop label
```

przy czym wartość wstawiana do rejestru cx powinna być dobrana doświadczalnie, tak by dźwięk, jaki się rozlega, był wystarczająco głośny. Na komputerze XT z zegarem 10 Mhz pętla opóźniająca nie była potrzebna.

Marek Ciężarek

```
ASSUME cs:code

code SEGMENT

        org     100h
Start:
        jmp     install

old9o   dw      0               ; Adres programu przerwania
old9s   dw      0               ; 9-tego w BIOS-ie.

click:
        push    ax
        push    cx
        in      al,61h
        xor     al,2
        out     61h,al          ; rusz membranę
        pushf
        call dword ptr cs:old9o ; skok do BIOS-u
        xor     al,2
        out     61h,al          ; rusz membranę
        pop     cx
        pop     ax
        iret
install:
        mov     ax,3509h        ; Pobranie adresu
        int     21h             ; przerwania 9
        mov     cs:old9o,bx     ; i zapamiętanie go
        mov     cs:old9s,es     ; na przyszłość.
        mov     dx,offset click ; Ustawienie nowego
        push    cs              ; adresu przerwania 9
        pop     ds              ; na początek procedury
        mov     ax,2509h        ; click.
        int     21h
        mov     dx,offset install ; Skończ, zostawiając
        int     27h             ; w pamięci wszystko przed
                                ; adresem install.

code ENDS

END Start
```

GMERANIE MIĘDZY KLAWISZAMI

Wbrew tytułowi nie będzie mowy o zastosowaniu komputerów w więziennictwie — słowo klawisz należy rozumieć dosłownie, w jego pierwotnym znaczeniu. Przyjrzymy się klawiaturze i sposobom jej komunikacji z naszymi programami.

Klawiatura w PC to na dobrą sprawę osobny, silnie wyspecjalizowany komputer. Jego sercem jest układ 8048 (w AT 8042), wykonujący kilka prostych, lecz istotnych zadań. Po pierwsze, po włączeniu komputera testuje klawiaturę, żeby wykryć jej ewentualne uszkodzenia. Po drugie, ilekroć zostanie naciśnięty (bądź puszczony) jakiś klawisz, powiadamia o tym komputer. Po trzecie wreszcie, gdy klawisz jest naciśnięty ponad pół sekundy, 8048 zaczyna wysyłać co jedną dziesiątą sekundy taki sygnał, jak po naciśnięciu klawisza. Te operacje, w połączeniu z oprogramowaniem zawartym w ROM BIOS-ie komputera, pozwalają na wczytywanie przez programy znaków z klawiatury.

Po naciśnięciu przez użytkownika dowolnego klawisza, układ 8048 żąda od procesora komputera wykonania przerwania 9, zajmującego się obsługą klawiatury. Program przerwania dziewiątego zaczyna od wczytania z portu 96 (60h) kodu przeszukiwania wciśniętego klawisza. Ten kod to nic innego, jak numer klawisza na klawiaturze — dotyczy to wszystkich klawiszy, bez wyjątku, łącznie z ctrl, alt, shift i PrtScr. Podobnie dzieje się po puszczeniu klawisza, z jedną istotną różnicą — do kodu przekazywanego do BIOS-u dodane jest wówczas 128, czyli — innymi słowy — ustawiony jest najstarszy bit przesyłanego bajtu. Działanie układu 8048 na tym się kończy, komputer już „wie", co się dzieje z klawiaturą.

Cała reszta, czyli tłumaczenie kodów przeszukiwania na odpowiednie znaki ASCII (lub kody pomocnicze, odpowiadające klawiszom dodatkowym — funkcyjnym i kierunkowym), odbywa się w sposób programowy, poza klawiaturą, w BIOS-ie. Nie jest to banalne zadanie, bo każdy klawisz może mieć — w zależności od tego, czy są naciśnięte któreś z klawiszy specjalnych (shift, alt, ctrl) — kilka znaczeń. Należy więc zapamiętać stan wszystkich klawiszy specjalnych, a także stan klawiatury, czyli czy aktywny jest NumLock, CapsLock lub ScrollLock. Również kody ASCII, wprowadzone są za pomocą kombinacji alt — klawisze klawiatury numerycznej, tłumaczone są w BIOS-ie. Kody ASCII i pomocnicze po przetłumaczeniu zapamiętywane są w obrębie zmiennych systemowych BIOS-u (0:041Eh), w ilości piętnastu sztuk, i na życzenie udostępniane potrzebującym ich programom.

Służy do tego celu przerwanie 22 (16h). (Warto podkreślić różnicę między przerwaniami 9 a 22 — pierwsze obsługuje klawiaturę, czyli urządzenie zewnętrzne, drugie obsługuje użytkownika. Nie należy ich ze sobą mylić, ale wiele osób to robi). Przerwanie 22 oferuje trzy różne programy usługowe, wybierane przez umieszczenie w rejestrze AH odpowiedniej liczby. Pozwalają one na uzyskanie najważniejszych informacji o klawiaturze i mogą stanowić podstawę komunikacji programu z użytkownikiem. Za ich pośrednictwem można otrzymać informacje o tym, czy był naciśnięty jakiś klawisz, sprowadzić jego kod (ASCII lub pomocniczy, o czym dalej) oraz sprawdzić stan klawiszy specjalnych. Krótki opis wszystkich trzech programów znajduje się w tabelce, więc nie będziemy im poświęcać zbyt wiele miejsca. Jest również druga możliwość — skorzystanie z usług DOS-u, który może pośredniczyć w dostępie programu do BIOS-u. Wybór drogi należy do programisty i wymaga nieco doświadczenia.

Stan klawiszy specjalnych zapamiętany jest w obszarze zmiennych BIOS-u pod adresem 0:0417h i właściwie można z niego korzystać bezpośrednio. O ile jednak IBM gwarantuje, że opis przerwania 16h nie ulegnie zmianie, nie można tego samego powiedzieć o podanym adresie, toteż lepiej zabezpieczyć się przed niespodziankami i korzystać z przerwania. Ta reguła — że korzystanie z przerwań jest bezpieczniejsze od bezpośredniego grzebania w pamięci — obowiązuje zresztą w odniesieniu do całego systemu, nie tylko klawiatury.

Teraz kilka słów o kodach pomocniczych. Odnoszą się one do klawiszy dodatkowych (funkcyjnych, kierunkowych) lub do kombinacji klawiszy z klawiszami specjalnymi (ctrl-D, alt-F1 itd.). Właściwie nie ma w nich żadnej logiki; poza niektórymi przypadkami nie sposób zapamiętać, jakiemu klawiszowi odpowiada jaki kod pomocniczy. Najlepiej spisać je sobie na kartce i trzymać gdzieś pod ręką lub postąpić tak, jak proponuję obok użytkownikom Turbo Pascala — przygotować zestaw stałych (za pomocą EQU w asemblerze i #define w C), do których można się potem odwoływać w swoich programach. Znacznie ułatwia to oprogramowywanie reakcji na klawisze dodatkowe.

Zamiast odwoływać się do dość prymitywnych programów BIOS-u, można skorzystać ze znacznie bardziej zaawansowanych usług oferowanych przez DOS. Wywołuje się je w podobny sposób jak programy usługowe BIOS-u — w rejestrze AH należy umieścić numer funkcji, po czym wykonać przerwanie 21h, czyli wywołać system operacyjny. Programów usługowych DOS-u, które pozwalają na wczytywanie z klawiatury pojedynczych znaków i nie wysyłają ich równocześnie na ekran, jest pięć. Ich opis znajduje się w tabelce. Za pośrednictwem DOS-u nie można się dowiedzieć niczego o stanie klawiszy specjalnych — gdy są nam potrzebne, trzeba korzystać z usług BIOS-u.

Z punktu widzenia naszych programów, dopóki korzystamy z pośrednictwa BIOS-u lub DOS-u, wszystkie rodzaje klawiatur, niezależnie od modelu (XT, AT) i klawiszy, zachowują się tak samo. Nie znaczy to jednak, że są one identyczne pod względem ukrytych w nich możliwości. Klawiatury do AT są znacznie bardziej skomplikowane i można je w pewnym zakresie programować, np. ustalić czas po jakim zacznie się powtarzanie klawisza i częstotliwość jego powtarzania. Można także ustawić stan diod LED (zgaszona, zapalona), czego nie da się zrobić w klawiaturze XT, a co często powoduje frustrację podczas korzystania z programów wymuszających ustawienie klawiatury numerycznej w odpowiadający im sposób (robi to na przykład SideKick w kalkulatorze).

Marcin Borkowski

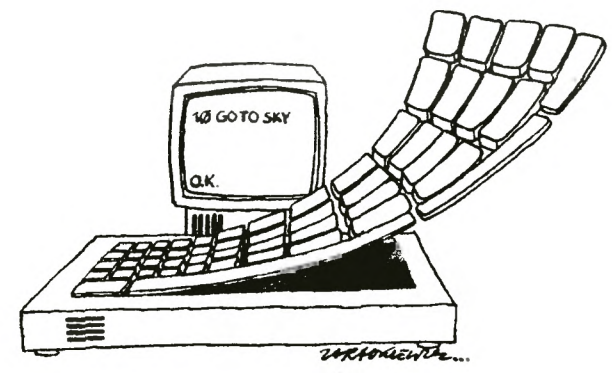
Korzystanie z klawiatury za pośrednictwem BIOS-u (INT 16h).

AH	Podjęta akcja
0	Odczyt następnego znaku. Jeśli znak był w buforze klawiatury, jest przekazywany od razu, w przeciwnym razie BIOS czeka na naciśnięcie klawisza. Po powrocie z przerwania - jeżeli bajt znajdujący się w AL jest różny od zera, to jest to kod ASCII naciśniętego klawisza. Jeśli AL=0, to w AH znajduje się kod pomocniczy.
1	Sprawdzenie stanu bufora klawiatury. Jeśli bufor jest pusty, wskaźnik zera (zero flag) jest ustawiony na 1, w przeciwnym wypadku jest równy 0.
2	Pobranie stanu klawiszy specjalnych. Bity w AL odpowiadają: (1 zawsze oznacza naciśnięty lub aktywny) 7 - Insert 6 - CapsLock 5 - NumLock 4 - ScrollLock 3 - alt 2 - ctrl 1 - lewy shift 0 - prawy shift

Korzystanie z klawiatury za pośrednictwem DOS-u (INT 21h).

AH	Podjęta akcja
06h	Bezpośrednia operacja wejścia-wyjścia. Jeżeli w rejestrze DL umieścimy 0FFh, to jeśli w buforze klawiatury znajduje się gotowy znak, zostanie wyzerowany znacznik zera, a kod klawisza będzie umieszczony w AL, jeśli bufor klawiatury był pusty będzie ustawiony znacznik zera. Jeżeli w rejestrze DL umieścimy inną liczbę niż 0FFh, odpowiadający jej znak zostanie wyświetlony na ekranie.
07h	Wczytanie znaku z klawiatury. Jeżeli w buforze klawiatury był gotowy znak, zostanie przekazany w rejestrze AL, w przeciwnym razie system czeka na naciśnięcie klawisza. Naciśnięcie ctrl-C nie daje żadnego efektu, toteż program nie może być przerwany bez wprowadzenia znaku z klawiatury.
08h	To samo co 07h, ale naciśnięcie ctrl-C powoduje wykonanie przez DOS INT 23h.
0Bh	Sprawdzenie czy w buforze klawiatury znajduje się jakiś znak. Jeżeli tak, w AL zwracana jest wartość 0FFh, w przeciwnym razie 0h.
0Ch	Wyczyszczenie bufora, i wywołanie jednej z funkcji 01h, 06h, 07h, 08h lub 0Ah. Numer funkcji, która ma być wywołana po wyczyszczeniu bufora, przekazuje się DOS-owi w rejestrze AL. Umieszczenie w AL innej niż wymienione wartości powoduje samo wyczyszczenie buforu, i powrót do programu.

Uwaga: przy wszystkich przerwaniach DOS-u służących do wczytywania znaków, wczytanie kodu klawisza dodatkowego odbywa się w dwóch etapach - po pierwszym wywołaniu DOS-u w AL zwracane jest zero, po drugim kod pomocniczy.



KLAWISZE JESZCZE RAZ

Podczas Gmerania w klawiszach mowa była (choć nie wprost) o programach asemblerowych, mogących w dowolnej chwili odwołać się do BIOS-u i przetestować wskaźnik zera.

W językach wyższego poziomu — takich jak C czy Pascal — odwołania takie wymagają nieco gimnastyki, toteż twórcy kompilatorów wśród funkcji bibliotecznych umieszczają zwykle przynajmniej dwie interesujące nas — jedną pozwalającą na sprawdzenie, czy w buforze klawiatury jest jakiś znak do wczytania (**keypressed**, **kbhit ()**), i drugą, wczytującą znak (**readkey, inkey$, getc ()** etc.).

Tak jest właśnie w przypadku Turbo Pascal-a — w module **crt** (lub w bibliotece systemowej w wersji kompilatora wcześniejszej niż 4.0) są zdefiniowane funkcje **keypressed** i **readkey**.

Nie korzystają one bezpośrednio z usług BIOS-u, sięgając do klawiatury za pośrednictwem przerwań DOS-u. Ma to dwie zalety — po pierwsze, sprawdzenie stanu buforu klawiatury nie wymaga testowania wskaźnika zera, lecz zawartości rejestru AL (DOS zwraca zero, gdy bufor jest pusty, i 255, gdy coś jest w buforze), po drugie, DOS zwraca kod klawisza w samym rejestrze AL, bez użycia rejestru AH. W przypadku klawiszy dodatkowych w rejeśtrze AL zwracana jest wartość zero, a kod pomocniczy pojawia się dopiero przy drugim wywołaniu DOS-u. Znacznie ułatwia to pisanie programów intensywnie korzystających z klawiszy dodatkowych.

```
unit keyboard;

interface

const
{   Kody klawiszy naciskanych razem z ctrl:     (bez _Null)    }
  _ctrlA = #1; _ctrlG =  #7; _ctrlM = #13; _ctrlS = #19; _ctrlY = #25;
  _ctrlB = #2; _ctrlH =  #8; _ctrlN = #14; _ctrlT = #20; _ctrlZ = #26;
  _ctrlC = #3; _ctrlI =  #9; _ctrlO = #15; _ctrlU = #21;
  _ctrlD = #4; _ctrlJ = #10; _ctrlP = #16; _ctrlV = #22;
  _ctrlE = #5; _ctrlK = #11; _ctrlQ = #17; _ctrlW = #23;
  _ctrlF = #6; _ctrlL = #12; _ctrlR = #18; _ctrlX = #24;
{   Kod specjalny, przed kodami rozszerzonymi:     }
  _Null   =   #0;
{   Kody klawiszy naciskanych razem z alt       (po _Null)    }
  _altA = #30;  _altG = #34;  _altM = #50;  _altS = #31;  _altY = #21;
  _altB = #48;  _altH = #35;  _altN = #49;  _altT = #20;  _altZ = #44;
  _altC = #46;  _altI = #23;  _altO = #24;  _altU = #22; _altmin = #130; {-}
  _altD = #32;  _altJ = #36; {_altP = #25;} _altV = #47; _altequ = #131; {=}
  _altE = #18;  _altK = #37; {_altQ = #16;} _altW = #17;
  _altF = #33;  _altL = #38;  _altR = #19;  _altX = #45;
  _alt1 = #120; _alt2 = #121; _alt3 = #122; _alt4 = #123; _alt5 = #124;
  _alt6 = #125; _alt7 = #126; _alt8 = #127; _alt9 = #128; _alt0 = #129;
{  Kody klawiszy sterujących kursorem:          (po _Null)    }
  _Up     =   #72;    _Down       =   #80;
  _Lft    =   #75;    _Rght       =   #77;
  _Ins    =   #82;    _Del        =   #83;
  _Home   =   #71;    _ctrlHome   =  #119;
  _End    =   #79;    _ctrlEnd    =  #117;
  _PgUp   =   #73;    _ctrlPgUp   =  #132;
  _PgDn   =   #81;    _ctrlPgDn   =  #118;
{  Kody klawiszy funkcyjnych:                          (po _Null)    }
  _F1   =  #59;    _shiftF1  =  #84;    _ctrlF1  =  #94;    _altF1  = #104;
  _F2   =  #60;    _shiftF2  =  #85;    _ctrlF2  =  #95;    _altF2  = #105;
  _F3   =  #61;    _shiftF3  =  #86;    _ctrlF3  =  #96;    _altF3  = #106;
  _F4   =  #62;    _shiftF4  =  #87;    _ctrlF4  =  #97;    _altF4  = #107;
  _F5   =  #63;    _shiftF5  =  #88;    _ctrlF5  =  #98;    _altF5  = #108;
  _F6   =  #64;    _shiftF6  =  #89;    _ctrlF6  =  #99;    _altF6  = #109;
  _F7   =  #65;    _shiftF7  =  #90;    _ctrlF7  = #100;    _altF7  = #110;
  _F8   =  #66;    _shiftF8  =  #91;    _ctrlF8  = #101;    _altF8  = #111;
  _F9   =  #67;    _shiftF9  =  #92;    _ctrlF9  = #102;    _altF9  = #112;
  _F10  =  #68;    _shiftF10 =  #93;    _ctrlF10 = #103;    _altF10 = #113;
{  Inne przydatne kody:  }
  _BS     =    #8;  _Tab    =    #9;  _esc    =   #27;
{  Funkcje i procedury - zestaw jest łatwy do rozszerzenia. }
function ReadKey : char;
  inline($b4/$08/$cd/$21);      { mov ah,08      int 21h  }
function KeyPressed : boolean;
  inline($b4/$0b/$cd/$21);      { mov ah,0Bh     int 21h  }
procedure wait_for_key;         { Czeka na naciśnięcie dowolnego klawisza.  }
procedure drop_a_key;           { Wczytuje znak z klawiatury i ignoruje go. }

implementation

procedure wait_for_key;
begin
  repeat until keypressed
end;

procedure drop_a_key;
begin
  if readkey=_Null then if boolean(readkey) then;
end;

end.
```

recenzje

WCIĄŻ NIE DOS-yć O DOS-ie

System, że jest — każdy widzi, gdy włączy komputer. System, jaki jest — widzi niewielu, a chciałby prawie każdy. Stąd od czasu do czasu ktoś pisze dla innych książkę o DOS-ie, w który wyposażone są nasze PC-ety.

Do tej pory wyszły trzy takie pozycje (nie wspominając o kilku broszurach), których znaczne części stanowią opisy systemu operacyjnego. W 1988 roku były to „Komputer osobisty typu IBM PC" Michała Kleibera i Romualda Szuniewicza i „IBM PC i PC DOS" Tomasza Kozdrowicza (recenzowaliśmy tę książkę w numerze 2/89), a w tym roku „System operacyjny MS-DOS" Jarosława Demineta.

Dwie pierwsze są napisane z myślą o początkujących użytkownikach, toteż sporo w nich informacji nie tylko o systemie, ale i o samym używaniu komputera. Mimo zmian, jakie nastąpiły w oferowanym na PC oprogramowaniu, książki te nie straciły nic na wartości, i można je z czystym sumieniem polecić każdemu, kto ma zamiar usiąść do pracy z komputerem, a nie robił tego nigdy wcześniej. (Książkę p. Kozdrowicza widziałem w sierpniu w Warszawie, można ją kupić bez większych problemów.) Trzecia pozycja, zgodnie z notką na czwartej stronie, przeznaczona jest dla szerokiego kręgu użytkowników mikrokomputerów osobistych. Pierwszy rzut oka nastawił mnie do niej przychylnie. Dalej bywało różnie.

Co jest w książce? Właściwie wszystko to, czego można się w niej spodziewać. Jest dość szeroki opis samej koncepcji systemu i możliwych sposobów jego wykorzystania. Jest opis poleceń (rezydentnych i zewnętrznych), ze składnią i wyliczeniem opcji. Jeden cały rozdział został poświęcony funkcjom systemowym, wywoływanym za pośrednictwem przerwań z poziomu asemblera. Jest opis formatu dysków, i różnych programów obsługujących urządzenia, a instalowanych za pośrednictwem zbioru konfiguracyjnego, a także opis PSP (Program Segment Prefix) zwykłych programów, tudzież sporo innych informacji. Zdawać by się więc mogło, że jest praktycznie wszystko, co może być potrzebne zaawansowanemu użytkownikowi.

Niestety, to bogactwo jest iluzoryczne. O ile można tę książkę przeczytać, po to by zdobyć przy jej pomocy dość szeroką wiedzę na temat systemu, o tyle korzystanie z niej w trakcie pracy, i traktowanie jako kompendium wiedzy o systemie, jest niemożliwe. Dzieje się tak z kilku powodów.

Po pierwsze, od czasu do czasu trafia się w książce na zupełnie nieoczekiwane braki. Opisany (i to precyzyjnie) jest FCB (File Control Block, blok opisu zbioru, odziedziczony po CP/M-ie), można go nawet znaleźć w indeksie. Konia z rzędem temu, kto szybko znajdzie informacje na temat DTA (Disk Transfer Adres) — nazwa ani skrót nie są w ogóle wymienione w tekście, choć opisane są oczywiście wszystkie funkcje związane z DTA. Również fiaskiem skończy się próba znalezienia informacji o zawartości nagłówka pliku typu *.exe — w poświęconym tym plikom rozdziale napisane jest co zawierają „między innymi" niektóre pozycje nagłówka — a że jest ich raptem czternaście, więc mogłyby być opisane precyzyjnie.

Po drugie, część przykładów zawartych w rozdziale poświęconym poleceniom systemu jest błędna — próby ich wykonania nie dają oczekiwanych efektów, choć nie sądzę by było tak z winy autora. Na stronie 93 znajduje się przykład dotyczący uruchamiania nowej kopii interpretatora poleceń — nie działający. Dwie strony wcześniej opisany jest filtr sort.exe — opcja +24 powodująca sortowanie według zawartości kolumny 24 i następnych powinna wyglądać /+24. Takie błędy podważają wiarygodność innych przykładów i opisów.

Po trzecie wreszcie, książka jest wyjątkowo niewygodna w użyciu. Śledziłem działanie pewnego programu, korzystając z odpluskwiacza. Ilekroć następowało odwołanie do systemu operacyjnego, usiłowałem je zidentyfikować, korzystając z rozdziału poświęconego funkcjom DOS-u. Za każdym razem trwało to przeraźliwie długo — nie ma spisu funkcji systemowych według numerów. Wszystkie funkcje są opisane, ale nie kolejno, tylko po pogrupowaniu ich w

```
program scredit;

uses keyboard,scrsave,graph;

label
   koniec;

var
   crd,md,xk,yk,dxy,oxk,oyk,fx,fy  :  integer;
   krzyzyk                         :  pointer;

procedure zmianaxy(var xy : integer;dxy,MaxXY : integer);
begin
   xy:=xy+dxy;        if xy<0 then xy:=0;     if xy>MaxXY then xy:=MaxXY
end;

begin
   crd:=0;  InitGraph(crd,md,'');
   line(0,5,10,5);    line(5,0,5,10);
   GetMem(krzyzyk,ImageSize(0,0,10,10));   GetImage(0,0,10,10,krzyzyk^);
   screenload(paramstr(1));
   oxk:=0; oyk:=0; xk:=0; yk:=0; dxy:=4;
   repeat
     PutImage(xk,yk,krzyzyk^,XORPut);
     case readkey of
       _esc  :  goto koniec;
       _Null :  case readkey of
                  _Home      :  begin xk:=0; yk:=0 end;
                  _End       :  begin xk:=GetMaxX-10; yk:=GetMaxY-10 end;
                  _PgUp      :  if dxy<64 then dxy:=dxy shl 1;
                  _PgDn      :  if dxy>1  then dxy:=dxy shr 1;
                  _ctrlPgUp  :  yk:=0;
                  _ctrlPgDn  :  yk:=GetMaxY-10;
                  _Up        :  zmianaxy(yk,-dxy,GetMaxY-10);
                  _Down      :  zmianaxy(yk,dxy,GetMaxY-10);
                  _Lft       :  zmianaxy(xk,-dxy,GetMaxX-10);
                  _Rght      :  zmianaxy(xk,dxy,GetMaxX-10);
                  _F1        :  PutPixel(xk+5,yk+5,0);
                  _F2        :  PutPixel(xk+5,yk+5,1);
                  _F3        :  begin fx:=xk+5; fy:=yk+5 end;
                  _F4        :  begin
                                  PutImage(xk,yk,krzyzyk^,XORPut);
                                  line(fx,fy,xk+5,yk+5);
                                  PutImage(xk,yk,krzyzyk^,XORPut)
                                end
                end  { of case }
     end;  { of case }
     PutImage(oxk,oyk,krzyzyk^,XORPut); oxk:=xk; oyk:=yk;
   until false;
 koniec:
   PutImage(xk,yk,krzyzyk^,XORPut);
   screensave(paramstr(1));
   closegraph;
end.
```

Ponieważ bardzo trudno jest zapamiętać wartości kodów pomocniczych odpowiadających poszczególnym klawiszom, aż prosi się o to, by zdefiniować odpowiednie stałe, z których można by korzystać we własnych programach. Dobrze by jednak było, by stałe te znalazły się w jednym module z funkcjami **readkey** i **keypressed**, co jest niemożliwe, dopóki mamy na myśli moduł **crt**. Nic jednak nie stoi na przeszkodzie, by przygotować własny moduł, zawierający obie wymienione funkcje, komplet potrzebnych stałych i kilka dodatkowych procedur. Proponuję nazwać ten moduł **keyboard**.

Jako przykład jego wykorzystania (i korzyści wynikających z jego używania) przygotowałem niewielki program, pozwalający na dokonywanie drobnych poprawek w przygotowanych wcześniej zbiorach graficznych (o ich przygotowaniu i module **scrsave** — karta Hercules — pisałem miesiąc temu). Używa się w tym celu klawiszy kierunkowych i czterech klawiszy funkcyjnych, ich funkcje są następujące:

strzałki — ruch kursora,
Home — kursor w lewy górny róg ekranu,
End — kursor w prawy dolny róg ekranu,
PgUp — zwiększenie skoku kursora,
PgDn — zmniejszenie skoku kursora,
^PgUp — skok kursora do góry ekranu, bez przesuwania w poziomie,
^PgDn — jak wyżej, ale na dół,
F1 — postawienie punktu w pozycji kursora,
F2 — skasowanie punktu w pozycji kursora,
F3 — zapamiętanie położenia kursora jako początku linii,
F4 — narysowanie linii — jej początek musi być wcześniej ustalony przy pomocy klawisza F3, koniec jest wskazywany przez kursor.

Program wywołuje się z jednym parametrem — nazwą zbioru zawierającego zapamiętany ekran.

Marcin Borkowski

---

bloki — dostęp do plików, zarządzanie pamięcią, znakowe wejście–wyjście itd. Pomysł jest ciekawy, ale zupełnie nie sprawdza się gdy chcemy mieć pod ręką wygodne źródło informacji o potrzebnych w danej chwili usługach proponowanych przez system. Podobną opinię wyraziło jeszcze kilka osób, więc wiem że nie jestem w tej sprawie odosobniony.

Dotychczas brak było na naszym rynku książki poświęconej DOS-owi a adresowanej do jego bardziej zaawansowanych użytkowników. Autor, pan Jarosław Deminet, trafił więc w lukę, ale jej nie wypełnił. Największą wadą tej książki jest to, że nie bardzo wiadomo do kogo jest adresowana. Poziom i zakres prezentowanych w niej informacji wskazują na zaawansowanych użytkowników, piszących własne programy, i korzystających (przynajmniej częściowo) z asemblera. Układ informacji uniemożliwia właśnie tej grupie efektywne korzystanie z książki w trakcie pracy. Szkoda.

Marcin Borkowski

Jarosław Deminet, „System operacyjny MS-DOS", Wydawnictwa Naukowo-Techniczne, Warszawa 1990, nakład 15 000, cena 15 000 zł.

BYTE

To co najlepsze w dziedzinie komputerów osobistych, z pewnością znajdzie się na łamach amerykańskiego miesięcznika „Byte", który w ciągu ostatnich piętnastu lat stał się największym i najpoważniejszym magazynem komputerowym świata.

Sierpniowy numer, który dotarł do naszej redakcji, przynosi między innymi wyniki ankiety czytelniczej na najlepszy produkt 1990 roku. Warto wiedzieć, że sama redakcja rokrocznie przyznaje własne, niezwykle cenione przez producentów nagrody. Tym razem głos oddano Czytelnikom. Za najlepszy mikroprocesor uznano **Intel 386** — stał się on podstawą całej rodziny komputerów umownie określanych „Trzysta osiemdziesiątki szóstki".

Królem wśród komputerów obwołali czytelnicy „Byte" ex aequo **Compaq Deskpro 386** i **Macintosha II** za szybkość i elegancję.

Najlepszą drukarką jest **Hewlett-Packard LaserJet II** — odznaczająca się trwałością i znakomitą jakością wykonania, z której słynie firma.

Komputer bez monitora nie jest wiele wart. Być może dlatego firma NEC, która produkuje monitor **MultiSync 3D**, ma powody do szczególnej radości za zwycięstwo swojego produktu w ankiecie Czytelników.

Podobnie Alan Shugart — szef firmy Seagate — może mówić o umocnieniu pozycji na rynku dzięki twardemu dyskowi **Seagate ST251**, który, jak głosowała większość, jest bezkonkurencyjny.

W dziedzinie sprzętu pozostali bohaterowie to karta graficzna **Paradise VGA Plus** firmy Western Digital, **Smartmodem 2400** firmy Hayes i **Microsoft Mouse**.

Oprogramowanie to osobna konkurencja, nie mniej ważna od sprzętu. O wartości komputera nie decyduje tylko szybkość przetwarzania danych, ale także jakość programów. Firmy softwarowe wprost prześcigają się w kokietowaniu użytkowników nie oszczędzając na reklamie, promocyjnych cenach, ulgach, specjalnej obsłudze itp. Użytkownicy umieją to docenić.

W tym roku za najlepszy edytor tekstów uznano **Word Perfect 5.1**, którego najpoważniejszym konkurentem był Microsoft Word.

Wśród arkuszy kalkulacyjnych od wielu lat bezkonkurencyjny jest **Lotus 1–2–3**. Tym razem jego wersja **2.2** została nagrodzona za jakość i „przyjazność" dla użytkownika.

Aldous Page Maker pokonał produkt firmy Rank Xerox Ventura Publisher w dziedzinie programów Desktop Publishing. Wśród baz danych program **Paradox 3** produkowany przez firmę Borland International Inc. okazał się lepszy od popularnego w Polsce dBASE III Plus i dBASE IV.

Za najlepszy kompilator uznano **Turbo Pascal 5**, wśród „użytków" umocnił swą pozycję **Norton Utilities Advanced Edition 3.X**, być może dzięki talentom Johna Sochy: wśród systemów operacyjnych od lat dzierży prym **MS-DOS**, w tym roku jego wersja **3.3**.

Wśród „Bardzo Ważnych Osób", celebrujących konkurs i piętnastolecie istnienia „Byte", byli między innymi Mitch Kapor — założyciel Lotus Development Corporation, Gary Kildall — Prezes Digital Research i Niklaus Wirth, któremu wdzięczni są wszyscy za języki: Pascal, Modula 2 i Oberon. Obsługę Dziennikarską zapewniali między innymi Stewart Alsop ze słynnej dziennikarskiej rodziny i Britt Hume z Washington Post.

Nas tam nie było ciałem, ale duchem przenieśliśmy się w miejsce, gdzie świętowano całą epokę w historii informatyki.

Bream

recenzje

NARC

PROGRAM DEARCHIWIZUJĄCY

Kiedy zaczyna brakować nam miejsca na dysku twardym lub na dyskietkach, sięgamy po programy kompresji plików.

Dobrze znanymi na naszym rynku są produkty firmy PKWARE INC. Program PKARC pozwala zaoszczędzić kilkadziesiąt procent pojemności dyskietki. Lepsze algorytmy dające jeszcze większy zysk zastosowano w programie PKZIP tej samej firmy. Oba narzędzia oferowane są w zestawach z programami rozpakowującymi — PKXARC i PKUNZIP. Niestety sposób korzystania z tych ostatnich nie jest specjalnie wygodny.

Znacznie łatwiejszy w obsłudze i posiadający większe możliwości jest program NARC. Jego autorem jest Gary Conway z firmy Infinity Design Concepts, Inc (USA). Program umożliwia pracę z plikami archiwizowanymi przez programy PKARC, PKPAK i PKZIP w dostępie swobodnym, w odróżnieniu od dostępu sekwencyjnego oferowanego przez produkty firmy PKWARE.

Po uruchomieniu program wyświetla wszystkie zarchiwizowane pliki znajdujące się w aktualnym katalogu, pozwalając na wybór jednego z nich do dalszej pracy (rys. 1). Jeśli nie ma takich plików (o rozszerzeniach ARC, ARK lub ZIP), to program pozwala na wybór nowego napędu lub podkatalogu.

Po wybraniu konkretnego pliku, który stanowi archiwum zawierające wiele spakowanych zbiorów, przechodzimy do menu przedstawionego na rys. 2. Istotnym jego elementem jest lista wszystkich zbiorów zawierająca pełne dane. Posługując się klawiszami sterujących ruchem kursora do góry i na dół jesteśmy w stanie wybrać interesujący nas plik. Możliwe jest rozpakowanie (EXTRACT), obejrzenie (VIEW), wydrukowanie (PRINT) lub skasowanie (KILL) każdego z nich. Dodatkowym ułatwieniem są operacje grupowe pozwalające na rozpakowanie większej liczby zbiorów jednocześnie.

Na szczególną uwagę zasługuje opcja VIEW, dzięki której możliwe jest nie tylko wygodne oglądanie pliku, bez konieczności jego rozpakowania, ale także wyszukiwanie zadanych napisów. Ogólnie biorąc program NARC jest znacznie wygodniejszy w obsłudze niż programy PK i różni się od nich w taki sam sposób, jak różnią się nakładki typu XTREE lub Norton Commander od „gołego" systemu operacyjnego.

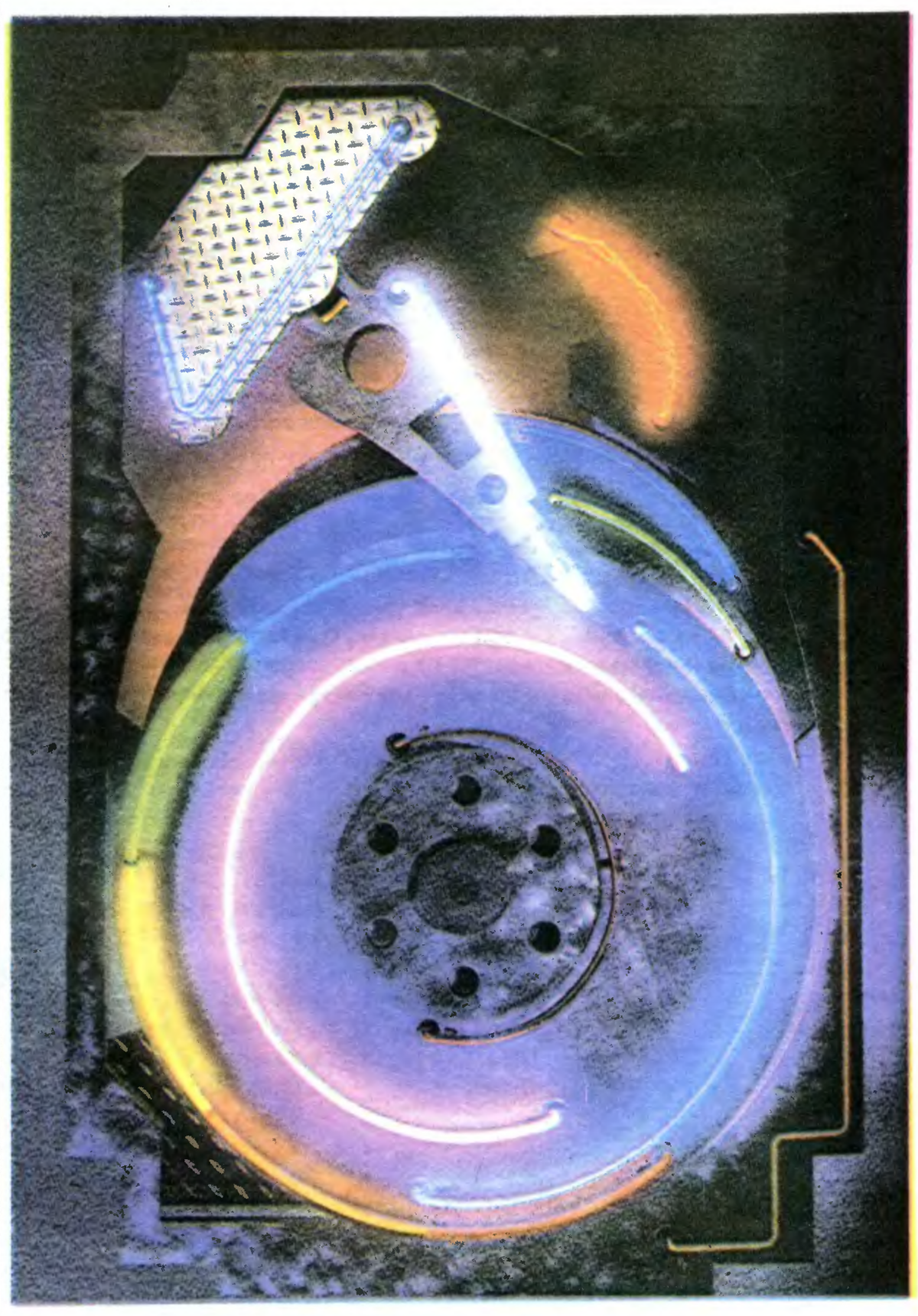
Jonasz Mayer

```
NARC zip  3.1   -   Pathname a:\
   Name        Length  Disk  Stowage  Ver  Stored Saved   Date      Time   CRC
============  =======  ====  =======  ===  ======= =====  ========  ======  =====

                                  Function   Keys
                        ENTER or Left Mouse   - select ZIP/ARC file
                        ESC or Right Mouse    - exits program
                        F5 - shell to DOS     S - change search spec.
                        F1 - HELP             O - toggle sort ON/OFF
AT       .ZIP
PRG-KNIG.ZIP
TOOLBOX  .ZIP
TUTOR    .ZIP
XT       .ZIP
```

Rys. 1 Menu wyboru pliku archiwizowanego.

```
NARC zip  3.1   -   Pathname a:\prg-knig.zip
   Name        Length  Disk  Stowage  Ver  Stored Saved   Date      Time   CRC
============  =======  ====  =======  ===  ======= =====  ========  ======  =====
VII-SYS .      61444   61k   Imploded  6    14111  78%   7 Oct 89  3:32a  454B
APP     .      12597   13k   Imploded  6     3478  73%   7 Oct 89  3:47a  C460
IV-SCR  .      27660   28k   Imploded  6     7433  74%   7 Oct 89  3:48a  1021
XII-SONY.      16950   17k   Imploded  6     3866  78%   7 Oct 89  4:17a  5C35
VIII-GSX.      37926   38k   Imploded  6     6414  84%   7 Oct 89  4:17a  A2A5
IX-WIND .      21027   21k   Imploded  6     5213  76%   7 Oct 89  4:32a  4D94
V-KBD   .       5285    6k   Imploded  6     1855  65%   7 Oct 89  4:33a  222B
XI-INT  .       4126    5k   Imploded  6     1501  64%   7 Oct 89  4:33a  C7BB
VI-PRI  .      23223   23k   Imploded  6     6500  73%   7 Oct 89  4:34a  E7B5
A-KBD   .       1122    2k   Imploded  6      427  62%   7 Oct 89  4:56a  4919
MORSE   .PAS     267    1k   Shrunk    1      200  26%   5 Dec 89  4:36p  AB58
VII-SYS .2      61414   60k   Imploded  6    14124  78%   5 Dec 89  8:15p  C109
PR      .PAS     3861    4k   Imploded  6     1221  69%   5 Dec 89  8:44p  7025
GRAPH   .SYS     6656    7k   Imploded  6     1962  71%                    285C
DEMO    .PAS     1920    2k   Imploded  6      732  62%                    4F3B
BITIMAGE.        7905    8k   Imploded  6     3654  54%   7 Oct 89  4:56a  412A

(c)87-89                                          Infinity Design Concepts
Totals   16     293383  296k                72691  76%     0 Tagged        0
                                F1 = HELP
  Extract      View       Print     ZIP-wind    Chg drive    DirTree      Quit
Extract Single File or Tagged Files.
```

Rys. 2 Menu operacji na wybranym pliku archiwizowanym.



CO? z obrotów dysków wynika dla zwykłego użytkownika

W jednym z popularnych polskich pism komputerowych (tytułu nie pamiętam, chyba zaczynał się na „B") czytałem ostatnio o pamięciach dyskowych. Przyznaję, że techniczna strona tych urządzeń wygląda korzystnie — nie wszystkie szczegóły zapamiętałem, ale takie np. gęstości upakowania głowic na minutę zrobiły na mnie wrażenie.

Jednak jako użytkownik, mający do rozwiązania konkretne problemy, zająłem się rozważaniami nad zastosowaniem tych cudów techniki do czegoś bardziej pożytecznego, niż samo tylko kręcenie się w kółko.

Na początek wydało się to zupełnie nierealne. Z punktu widzenia użytkownika, którego programy działają w jednostce centralnej komputera, pamięć dyskowa jawi się jako coś w rodzaju magicznej skrzynki, do której można wrzucać zawartość fragmentu pamięci operacyjnej. Magia skrzynki polega na tym, że w przyszłości — za pięć sekund albo za pięć lat — mogę to, co wrzuciłem, otrzymać z powrotem do PaO. Proste i przyjemne, ale tylko na poziomie koncepcji. Diabeł siedzi w szczegółach. W naszym przypadku — w szczegółach organizacji przesyłania. Aby transmisja porcji danych doszła do skutku, trzeba najpierw ustalić i przekazać parametry: adres i długość obszaru w PaO, z (lub do) którego przesyłane będą dane i to samo dla obszaru dyskowego (a więc niestety muszę znać się na sektorach i ścieżkach, nie mogę o nich zapomnieć). Gdy parametry przygotowane, można wydać rozkaz wykonania transmisji, ale to nie koniec, bo trzeba poczekać, aż urządzenie dyskowe prześle odpowiedź, czy operacja została wykonana pomyślnie, a jeśli nie, to jakie błędy wystąpiły. Rozsądek podpowiada jeszcze, że nie wszystkie operacje, wykonalne od strony technicznej, będą miały sens użytkowy. Np. dla urządzenia dyskowego polecenie zapisania do sektora, w którym już są dane jest zupełnie poprawne — w jego wyniku poprzednie dane zostają zatarte, a na ich miejscu są wpisane nowe. Dodatkowo więc muszę śledzić, które fragmenty dysku są już zajęte, aby nie zniszczyć żadnych potrzebnych informacji.

Spore wymagania, a ja mam własne poważne problemy do rozwiązania przy pomocy komputera, i miałem nadzieję, że pamięć dyskowa uprości ich rozwiązanie, a nie będzie stawiać mi dodatkowych zadań.

Z drugiej strony, trudno nie przyjmować do wiadomości, że aby transmisja mogła być wykonana, jej parametry muszą najpierw zostać odpowiednio ustawione. Takie są fakty, a o faktach się nie dyskutuje. Rozwiązaniem godzącym konieczność żmudnej obsługi z wymaganiami użytkownika, który chce się skoncentrować na własnych problemach (rozwiązaniem niewątpliwie udanym, bo bardzo rozpowszechnionym), jest **zbiór**.

Trochę szczegółów na temat realizacji tej koncepcji też znalazłem we wspomnianych artykułach, generalnie dotarło do mnie, że jeśli chcę przechować na dysku porcję danych, to wystarczy, abym nadał tej porcji jakąś nazwę, a specjalne programy, stanowiące część wyposażenia maszyny, zajmą się za mnie wszystkimi szczegółami technicznymi transmisji. Tę propozycję odebrałem jako projekt nowej magicznej skrzynki: wrzucam do niej jakąś logicznie zamkniętą porcję danych i nazwę, która będzie mi się z tą porcją danych kojarzyć. Jeśli w przyszłości — za pięć minut lub pięć tygodni — wrzucę znowu do skrzynki tę nazwę, to skrzynka odda mi moje dane. Autorzy tamtego artykułu, (panowie Andrzej i Krul) przestrzegali mnie jednak, że rozwiązanie to, choć na pewno znacznie ogranicza moje kłopoty, to może samo stać się przyczyną nowych kłopotów. Szczególnie przy przerwaniu pracy komputera w trakcie pisania do zbioru, wszystko na dysku może się tak pięknie pozająckować, że ani system operacyjny, ani żaden inny się w tym nie połapie*).

No cóż, trudno, będę musiał uważać i unikać takiego niezdrowego przerywania, ale z korzystania ze zbiorów nie zrezygnuję. Zawsze gdy trzeba przechować jakąś jednolitą porcję informacji, którą mogę przekazać systemowi jako całość, zbiory uwolnią mnie od kłopotów. A takich okazji powinno być sporo, bo wszystkie moje programy mogą być traktowane w ten sposób.

Niestety, moje zadania wymagają również możliwości dobierania się do fragmentów danych. Zresztą przyszła już najwyższa pora, aby zdradzić, jakie zadanie mam zamiar rozwiązać. Otóż chcę utrzymywać na dysku aktualną książkę telefoniczną. Będą w niej zawarci wszyscy abonenci sporego miasta, więc nie uda się zmieścić całych danych równocześnie w pamięci operacyjnej. Nie mogę więc przerzucać kompletu swoich danych ze zbioru na dysku do PaO, dokonywać poprawek i na koniec zapisywać uaktualnionej wersji w całości z powrotem do zbioru. A aktualizacji będzie sporo, gdyż codziennie abonenci są odłączani, ich miejsce zajmują nowi, wielu zmienia adres, nazwisko, nazwę.

W takim razie nie ma innego wyjścia, tylko pracować na fragmentach zbioru danych. Żadna z dwu poprzednich „skrzynek pamiętających" nie wydaje się być tu rewelacyjna (choć każdą z nich można by się od biedy posłużyć). Żeby sprawnie rozwiązać moje zadanie, przydałoby się urządzenie, do którego mógłbym wrzucać z pamięci operacyjnej komplet informacji opisujących jednego abonenta i które później wyrzuci z powrotem wybrany zapis. Jeśli będę miał taką skrzynkę lub innymi słowy, jeśli uda się tak zorganizować dostęp do pamięci dyskowej, to mogę zupełnie zapomnieć o ścieżkach, sektorach i buforach. Będę uważał, że mój zbiór danych zbudowany jest z zapisów, lub inaczej mówiąc z **rekordów logicznych** **).

Mówimy o rekordzie logicznym, gdyż jest on wyznaczony (jego wielkość i budowa są wyznaczone) przez interpretację danych, przez ich sens czy też logiczną konstrukcję. W mojej książce telefonicznej w skład rekordu logicznego ***) wchodzą:

nazwisko, imię, numer telefonu, ulica, numer domu.

I jeszcze jedna ważna rzecz: chciałem, aby moja nowa skrzynka mogła podać mi do pamięci operacyjnej treść ***wybranego*** zapisu, ale nie powiedziałem, jak będę go wybierał! Inaczej, w jaki sposób mam wskazać, o który zapis mi chodzi. Żeby to rozwiązać, muszę wyróżnić jedną z danych w rekordzie (jedno z pól) jako identyfikator. Aby dostać z dysku zawartość rekordu, będę podawał treść identyfikatora.

Korzystanie ze zbiorów o strukturze zdefiniowanej na potrzeby konkretnego zadania jest bardzo wygodne, ale na razie możemy o tym mówić tylko w trybie warunkowym — magiczna skrzynka trzeciego rzędu jest tylko pobożnym życzeniem. Czy więc możemy mieć ją naprawdę? Ależ oczywiście, i to przynajmniej na dwa sposoby, tak jak zresztą większość rzeczy: albo trzeba kupić gotową, albo zrobić samemu.

Zwykle maszyny oferowane do przetwarzania danych mają gotowe oprogramowanie pozwalające użytkownikowi definiować i wykorzystywać zbiory w/g własnych wymagań. Rozpowszechnione jest dołączanie do translatora języka programowania standardowych procedur operacji na zbiorach, które można zwyczajnie wywoływać w programach użytkowych. Procedury te załatwiają wymianę danych między jakąś (obsługiwaną przez system operacyjny) pamięcią dyskową a obiektami dostępnymi w danym języku programowania, np. tablicami czy zmiennymi. Reszta szczegółów opisana jest w dokumentacji, którą kupujący otrzymuje wraz z oprogramowaniem.

Aby zrobić samemu skrzynkę trzeciego rodzaju, należy najpierw chwilę pomyśleć. Można wtedy zauważyć, że najprawdopodobniej nie musimy zaczynać od najniższego poziomu obsługi transmisji, jeśli tylko mamy system operacyjny, który operuje na zbiorach i który daje możliwość podłączenia się do tych operacji.

Przecież system przesyłając zbiory robi to fragmentami — fragmenty te nazywane są rekordami fizycznymi ****). Jeśli uda nam się nawiązać z nim porozumienie, to uzyskamy możliwość wczytywania i zapisu rekordów fizycznych, bez martwienia się o adresy fizyczne i zajęte obszary. Teraz trzeba tylko założyć sobie jakiś skorowidz rekordów logicznych. Może to być np. tablica zawierająca dla każdego zapisu dwie informacje: (identyfikator rekordu logicznego i nr rekordu fizycznego zawierającego ten rekord logiczny). Tablicę tę przechowujemy oczywiście również na dysku wraz z treścią zapisów. Jeśli się zmieści, to rozpoczynając pracę możemy wczytać ją całą do PaO, jeśli nie, to będziemy musieli podzielić ją na fragmenty. W jednym takim fragmencie można np. mieć skorowidz wszystkich rekordów zaczynających się na tę samą literę — wszystko zależy od przewidywanej liczby danych. Ot i całe wnętrze naszej skrzynki. Jej działanie jest również nieskomplikowane. Gdy użytkownik żąda odszukania danych o podanym nazwisku, najpierw znajdujemy nazwisko w skorowidzu (może to wymagać odczytania z dysku jednego lub więcej fragmentów skorowidza), następnie odczytujemy numer rekordu fizycznego, odczytujemy ten rekord, wybieramy z niego odpowiedni fragment i przekazujemy użytkownikowi.

Nasze oprogramowanie widziane w powyższy sposób składa się z pewnych warstw. Najpierw mamy sprzęt i jego fizyczne możliwości. Pierwsza warstwa oprogramowania systemowego pozwala korzystać już z nieco wygodniejszych obiektów — fizycznych bloków grupowanych w zbiory. Następna warstwa daje do dyspozycji rekordy logiczne, dopasowane budową do zastosowania. Opierając się na jej możliwościach mogę stworzyć następną warstwę: program użytkowy, który pozwoli na wpisanie, aktualizację i wydruk książki telefonicznej. Każda z warstw może być widziana z góry (z następnej, wyższej warstwy) jako pewnego rodzaju skrzynka, z której można korzystać nie wnikając w szczegóły zastosowanych rozwiązań.

Z kolei realizacja każdej z warstw stwarza własne, bardzo interesujące problemy. Niespodziewanie wiele problemów może pojawić się również w najwyższej warstwie, tej, w której obracam się jako użytkownik. Ale to już całkiem inna historia.

---

*) I choćby przyszło tysiąc systemów, też nie rozwiążą takich problemów.

**) Słowo „rekord" w polskim języku informatycznym pochodzi od angielskiego „record", znaczącego właśnie zapis.

***) Zwykle elementarne jednostki danych, z których budowany jest rekord nazywamy **polami**.

****) Używana jest również nazwa „blok"

Andy Crool®

WSZYSTKO DLA WSZYSTKICH

ATASERW

43-100 TYCHY
ul. Lencewicza 46/3
tel. 27 69 66

oferuje świetne rozwiązania sprzętowe

do ATARI XL/XE:

1. TURBO DOS — wspaniały DOS na kartridżu
2. TOP DRIVE — do stacji 1050, LDW 2000, CALIFORNIA samodzielny montaż — (rec. INFORMIK III/88)
3. INTERFEJS CENTRONIKS
4. ROZSZERZENIA PAMIĘCI
5. BASIC XE — kartridż
6. TUR DOS+BUG65+MAC65 –kartridż

12 miesięcy gwarancji. Informacje i zamówienia telefonicznie (wtorek 8–12, środa, czwartek (16–18) i listownie po otrzymaniu koperty zwrotnej.

B 16

Atari Turbo 2000 F

Nowy system transmisji danych z magnetofonem przyspieszony do 6700 bodów.

Komplet:
— cartridge
— oprogramowanie
— przeróbka magnetofonu
— instrukcja obsługi
— 12 miesięcy gwarancji

Instalacje wykonujemy na poczekaniu.

Interfejs do zwykłego magnetofonu

Duży wybór oprogramowania w standardzie TURBO–2000.

Informacja:
Tel. 33-40-91
Korespondencja:
MUEL ul. Cząstkowska 30,
01-678 Warszawa.

B-31

MÓZG PROCESOR!

to rewelacyjna polska gra przygodowa firmy

COMPUTER ADVENTURE STUDIO dla Atari XL/XE (taśma + opis), dla Spectrum, Timex, Junior (taśma + opis), — dla Atari XL/XE (dyskietka + opis). Cena zestawu — równowartość 2,5 USD. Test w „Bajtku nr 10/89.

Zamówienia prosimy kierować:

COMPUTER ADVENTURE STUDIO

32-700 Bochnia, ul.
Kazimierza Wielkiego 37/45
tel. (o-197) 242-47 8–16

U W A G A !
Nawiążemy współpracę z autorami orginalnych polskich programów i scenariuszy.

B 4

Programy na ZX Spectrum tanio!
Jan Kondraciuk
ul. Słowackiego 25/60, 05—120 Legionowo B 66

Oficjalny dystrybutor oryginalnego, polskiego oprogramowania do komputerów Spectrum, Timex, Atari, Commodore, IBM oferuje gry i programy użytkowe sklepom, klubom, studiom komputerowym i wypożyczalniom.
Poszukujemy osób zainteresowanych współpracą w rozpowszechnianiu oprogramowania.
Przyjmujemy zlecenia na pisanie programów.
Nasze programy reprezentują profesjonalny poziom wykonania oraz posiadają ochronę prawną. Najlepsze z nich mają zapewnioną reklamę w szerokim kręgu odbiorców.

SPEKTRA
21-422 Stanin
tel. 11—70

B 65

joy

wysyłka natychmiastowa za zaliczeniem pocztowym

JOYSTICKI do Atari, Commodore Spectrum, Amstrad, „Kable z wtyczką do joysticka"
Precyzyjny mechanizm, specjalne styki.
6 miesięcy gwarancja. Interface do Spectrum.

ELEKTROMECHANIKA
ul. Cegielniana 17
32—410 Dobczyce

B 9

Zakład Usług Elektronicznych „HOMECOMP"

(do niedawna AZUSPHW) poleca usługi w zakresie serwisu komputerów: Spectrum, C-64, C+4, Timex, Atari oraz zasilaczy komputerowych.
Warszawa ul. Puławska 102,
tel. 44-87-89
czynny w godz. 11 — 19, rachunki, gwarancja.

B 63

SAM WYKONASZ OBWODY DRUKOWANE

Zestaw (laminat, odczynniki, instrukcja)
Cena 3550 zł. plus opłaty pocztowe.
Wysyłka za zaliczeniem pocztowym.
Zamówienia kierować: A. Kawczyński 90–950 Łódź-1 skrytka pocztowa 344. Płatne przy odbiorze paczki.
ZAWSZE AKTUALNE!

B 11

ATARI, COMMODORE

Układy scalone, inne części zamienne, instrukcje serwisowe, itp.
Rozszerzenia pamięci do Amiga 500.
TANIO! Np. 6581-160 tys. zł., 6569--180 tys. zł. Freddie — 160 tys. zł.
Informacje po otrzymaniu koperty zwrotnej.

INTER BAZAR,
Os. Centrum 1
33-170 Tuchow, tel. (014525) 534 **B 57**

Atari XL/XE
STUDIO KOMPUTEROWE

MIKROBIT

oferuje:

1. KSO TURBO 2000
Rewelacyjny kasetowy system do samodzielnego montażu w magnetofonach firmowych.
Podstawowe zalety:
— oszczędność kaset (C60 — około 40 programów)
— 10-krotne przyspieszenie wczytywania
— wysoka wgrywalność
— system operacyjny na Cartridgeu
— możliwość kopiowania Standard, — Turbo — Turbo, Standard
— współpraca z BASIC-em
— wyjątkowa prostota montażu.

2. Bogaty wybór oprogramowania
w systemie KSO Turbo 2000 oraz na dyskietkach.

3. Interfejsy CENTRONIC'S.
Sprzedaż wysyłkowa, gwarancja

Adres: Studio MIKROBIT
ul. Malborska 6/160
03—286 Warszawa

ATARI ST
Programy użytkowe, gry (również nowości)
oraz instrukcje wysyłam pocztą po atrakcyjnych cenach.
ul. Krasiczyńska 5/92
03—379 Warszawa

B 54

„BETA B"

AGENCJA INFORMATYCZNA
41-200 Sosnowiec,
skrytka 254
Telef. 632-935 690-385
oferuje również wysyłkowo:
Programy, Instrukcje, Literaturę dla komputerów ACORN AMSTRAD ATARI COMMODORE SHARP IBM

B 18

DYSKI

3 1/2" 7100 zł, 5 1/4 3200 zł.
Gwarantowana „life-long" jakość.

RABATY!

Szybka dostawa. Bezwarunkowe reklamacje.
Sprzedaż wysyłkowa i informacje: skr. 157, 59-300 LUBIN, woj. Legnickie

B 64

Elementy Elektroniczne — Skup i Sprzedaż szeroki asortyment — umiarkowane ceny Przyjmę zamówienia od osób prywatnych i instytucji (rychunki). Prowadzę sprzedaż wysyłkową.
Oferta — koperta zwrotna + znaczek.
Lesław Buras, 51-639 Wrocław ul. Wyczółkowskiego 17

SUPER OKAZJA!!
Tylko dla znawcow
Komputer produkcji amerykanskiej
Z O R B A
CP/M, monitor 9"
2*FDD 5.25"
+oprogramowanie
SPRZEDAM
tel.42-82-59 W-wa

ZX SPECTRUM ATARI system turbo, TIMEX FDD 3000,

programy użytkowe, edukacyjne, gry, instrukcje, podręczniki
wysyłka na cały kraj
rachunki
informacje po nadesłaniu koperty + znaczek.

2"P.K.T.S." Studio Komputerowe
00-103 Warszawa
ul. Królewska 43 m 25

B52

AMIGA, ATARI ST gry

programy użytkowe, literatura
Studio Komuterowe
„AMIWARE"
00—851 Warszawa,
ul. Waliców 20/1516
Wysyłka na cały kraj, katalog gratis

B 61

SOUND — trójkanałowy, stereofoniczny,
przelotowy interfejs muzyczny (AY-3-8910) do ZX Spectrum i Timexa.
Możliwa wysyłka pocztą.
„DYMAREX"
ul. Meissnera 14 m1
03—982 Warszawa, tel. 15-93-38
godz. 18—20

B 58

Programy C-16, C-116, C plus/4, C-64 wysyłam pocztą.
Katalog gratis po otrzymaniu koperty zwrotnej.
Nagrywanie Programów Komputerowych
TADEUSZ MIECZKOWSKI
ul. Lenina 104/3
58—304 Wałbrzych

B 62

KOMPUTER — SERVICE

Naprawa komputerów
COMMODORE, IBM, SPECTRUM, TIMEX
oraz
SERWIS i przeróbki
zasilacze monitorów, drukarek.
Instalacje polskich znaków.
Kraków, tel. (012)33-96-51
poniedziałek — piątek
godz. 10.00 – 13.00,
20.00 – 21.00 B 15

ATARI 800 XL, 65 XE, 130 XE

Sprzedaż wysyłkowa
gier i programów użytkowych
na kasetach i dyskietkach.
Również w systemie TURBO 2000
Wszystkie nowości!!!
Instrukcje i literatura.
Dla zainteresowanych rachunki.

ANWIKOL

03-721 Warszawa ul. Jagiellońska 3/28.

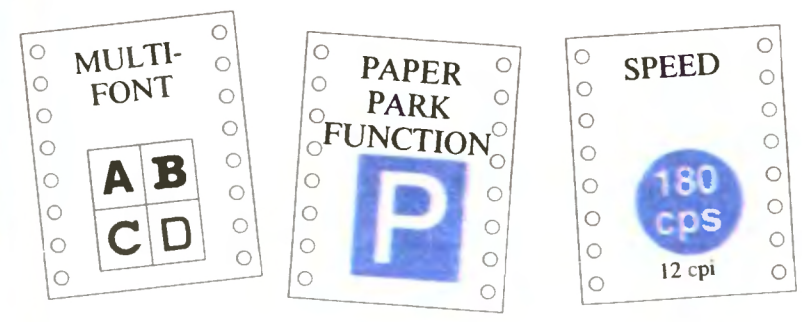
(SB 74)

DRUKARKA STAR LC-20

— to nowa, szybsza LC-10

- Prędkość druku: 180 zn./sek.
- Jakość druku: standard oraz NLQ
- Traktor pchający
- „Parkowanie" papieru
- Automatyka oddzierania papieru
- Interfejs Centronics

Cena 2.500.000 (orientacyjna cena detaliczna)

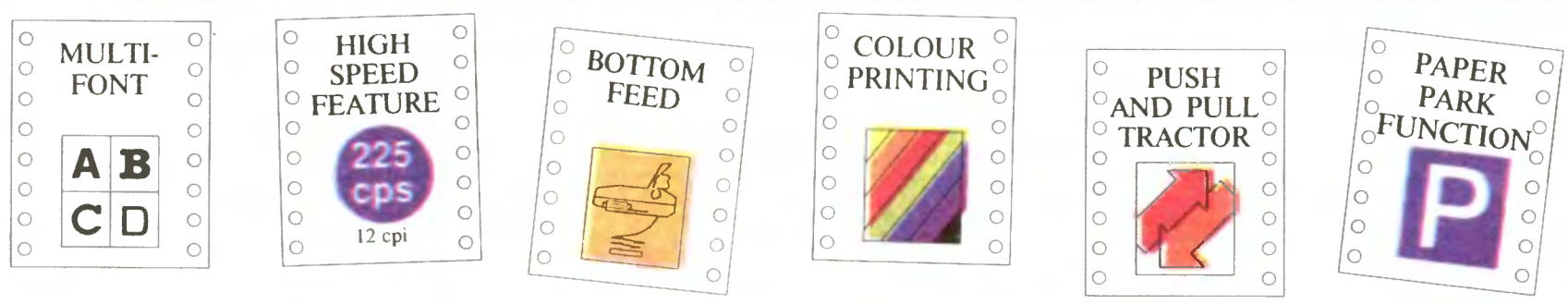


DRUKARKA LC-200

— Star znów ustanawia nowy standard!

- Max. prędkość druku: 225 zn./sek.
- Druk kolorowy
- Możliwość podawania papieru od dołu
- Traktor pchający i ciągnący
- „Parkowanie" papieru
- Automatyka oddzierania papieru
- Interfejs Centronics

Cena 3.900.000
(orientacyjna cena detaliczna)



star
Twoja drukarka

Przedstawicielstwo w Polsce
ABC Data Warszawa
ul. Waliców 13

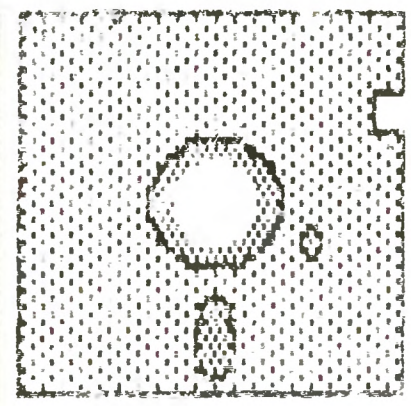
tel. 24-11-43
24-78-35
tx. 816-423

ABC Data

Ceny na dzień 11.09.1990

	Giełda	Sklep	Pewex	Zachód
	tys. zł		$	
SINCLAIR				
ZX 81	200	—	—	—
ZX Spectrum 48	850	950	—	100
ZX Spectrum +	900	1100	—	120
ZX Spectrum + 2	1350	—	—	160
Timex 2048	1100	1200	—	—
stacja FDD3000	900	1300	—	—
stacja FDD3	700	—	—	—
drukarka GP–50	—	500	—	—
Masterface I	100	—	—	—
AY 3–8910	130	—	—	—
COMMODORE				
C 64	1400	1700	199	200
C 64 Desk–Top	2000	—	—	—
C 128	2200	2300	—	260
C 128 D	4100	3900	599	360
Amiga 500	5100	5800	765	700
Amiga 2000	17000	—	—	1800
magnetofon	280	290	30	—
stacja 1541	1400	1700	—	—
stacja 1571	1800	2200	—	230
stacja Oceanic	1400	—	—	130
LC 10C	2200	2500	299	260
Final II	100	—	—	—
Final III	200	—	—	—
Action V	300	—	—	—
ATARI				
800 XL	1000	1100	—	—
65 XE	1100	1200	127	—
130 XE	1400	1550	199	190
520 STFM	4000	4300	499	350
1040 STFM	—	—	899	850
magnetofon	350	350	51	—
stacja 1050	1200	1400	—	140
monitor SM124	2000	—	199	—
monitor SM1224	—	—	479	480
drukarka 1029	800	1300	—	—
Turbo 2000	100	100	—	—
Centronics	180	190	—	—
AMSTRAD				
464	2000	1700	—	—
664	—	2200	—	—
6128	2800	2900	—	330
PCW 8256	—	—	—	—
PCW 8512	—	5100	—	—
PC 1512	8000	—	—	650
PC 1640	12000	—	—	1150
IBM				
PC XT stand.	7000	7400	499	500
PC AT stand.	—	—	—	750
HD 20 MB	3000	2800	370	220
napęd 5''	600	—	100	60
monitor amber	—	—	219	150
klawiatura	700	380	—	180
INNE				
dyskietki 5''	3.3–7	4.5–10	—	0.3–3
dyskietki 3.5''	8–13	9–12	2	1–3
dyskietki 3''	19–26	20	—	2–4
kasety C–60	12	—	1	1
monitor Neptun	600	680	—	—
joystick	50–80	60–90	7–9	10–30

Sklepy „Bajtka": Bytom, ul. Koniewa 6, tel. (832) 81-57-01
Kraków, ul. Pstrowskiego, tel. (012) 56-54-52



INDYWIDUALNY BANK DANYCH

Maciej Romanowski lat 18. Posiada Amstrada CPC 464 oraz stację dysków 3" (CPM 2.2). Oprogramowanie: 100 gier i kilka programów użytkowych. Zainteresowanie: elektronika i informatyka oraz muzyka komputerowa. Proponuje wymianę doświadczeń i oprogramowania.
Adres: ul. Miedziana 1/29, 82-500 Kwidzyń.

Dariusz Beta lat 13, posiada komputer VZ 200. Poszukuje programów użytkowych oraz gier.
Adres: 44-224 Knurów, ul. K.R.N. 2a/11

Janusz Sierżęga lat 17. Posiada Atari 800XL, magnetofon XC12 działający w systemie AST, 2000, ATT, TSF. Oprogramowanie około 100 gier i programów użytkowych. Prosi o kontakt z posiadaczami magnetofonów działających w systemie Turbo w celu wymiany programów i gier.
Adres: 36-004 Łąka 542, woj. Rzeszów

Grzegorz Fior lat 12, posiada komputer Unipolbrit 2086 oraz magnetofon. Oprogramowanie: około 400 gier i programy użytkowe. Proponuje wymianę oprogramowania. Odpowie na każdy list.
Adres: 48-310 Złotogłowice 12, g. Nysa

Krzysztof Dęga — kucharz. Posiada ZX Spectrum + joystick Suguz II, trójkanałowy generator dźwięku (AY 3--8910). Proponuje wymianę programów muzycznych na AY.
Adres: ul. Bydgoska 61 c/3, 64-920 Piła

Cezary Jakimik lat 15. Posiada Atari 65XE wraz z monitorem i magnetofonem XC12 pracującym w systemie Turbo 2000F. Proponuję wymianę oprogramowania z posiadaczami tego systemu.
Adres: 16-100 Sokółka, Oś. Centrum 19/6

Maciej i Piotr Kwapulińscy lat 15. Posiadają Commodore 64, magnetofon Datasette 1530. Oprogramowanie: 150 nowych gier i programów użytkowych. Nawiążą kontakt w celu wymiany oprogramowania.
Adres: 84-120 Władysławowo, ul. Spokojna 10.

Michał Matuszczyk, lat 15. Posiada ZX Spectrum i C-64 z magnetofonem i turbo. Dysponuje wieloma grami i programami użytkowymi na Spectrum 250, C-64, 500. Pragnie wymienić literaturę i programy.
Adres: 42-718 Kochcice, Zamek 2

Piotr Kreglicki lat 15. Posiada komputer IBM PC XT z drukarką Epson FX 800. Pragnie nawiązać kontakt w celu wymiany oprogramowania i literatury.
Adres: 21-040 Świdnik, ul. Skarzyńskiego 5/51

Rafał Chrząszcz, lat 15. Posiada Commodore 64c z magnetofonem 1530, X-soft cartridge, oraz ok. 1900 programów, gier (same nowości). Proponuje wymianę oprogramowania i gier.
Adres: ul. Nickla 42/9, 41-908 Bytom 8 — Miechowice.

Wojtek Trzaska lat 15. Posiada Amigę 500 z 1MB pamięci i drukarką. Pragnie nawiązać kontakt w celu wymiany oprogramowania i doświadczeń.
Adres: 41-800 Zabrze ul. Łokietka 5/3

Grzegorz Swietlicki lat 16. Posiada Commodore C 64C z magnetofonem Datasette 1531. Pragnie nawiązać kontakt w celu wymiany oprogramowania.
Adres: 21-040 Świdnik ul. Skarżyńskiego 7/16

Na listy czytelników odpowiadają autorzy „Bajtka"

Drogi Bajtku!

Czy lepiej kupić stację dysków dwukieszeniową, czy może jednokieszeniową (FDD 3000)? Jakie korzyści są z posiadania dwóch kieszeni?

Tomasz Lewandowicz, Łódź

Stacja z dwoma kieszeniami jest nie tylko droższa, lecz także lepsza. Druga kieszeń ułatwia kopiowanie programów, obsługę plików i wszystko to, co wynika z zarządzania większym zbiorem danych (2∗140KB dla standardowego TOS-u). Na przykład tworzymy własny program, którego kolejne wersje przechowujemy na dysku w napędzie B. W napędzie A możemy mieć dysk z wszystkimi programami użytkowymi, jak: monitory, edytory, asembler lub inne, które to programy wykorzystujemy do modyfikowania naszego tworzonego programu.

Dzięki strukturze TOS-u w bardzo prosty sposób możemy poruszać się między dwoma napędami, co w radykalnym stopniu zwiększa elastyczność systemu (przekonali się o tym ci, którzy do stacji jednokieszeniowej dokupili drugi napęd).

Jeśli jednak mamy zamiar kupić stację dwukieszeniową, należy wiedzieć, że wtedy podłączenie kolejnego (trzeciego; np. 5.25 ~) napędu nie będzie proste. Trzeci napęd wymaga już oddzielnego zasilania i dodatkowego łącza z kontrolerem.

Czy program SOUND TRACKER tworzy muzyczne programy demonstracyjne, tzw. dema?

Tomek Pokorny, Radzymin

Pytanie to wynikło chyba z błędnej interpretacji recenzji programu zamieszczonej w „Bajtku" 5/6 '90.

Program SOUND TRACKER jest profesjonalnym narzędziem do tworzenia muzyki. Zachodzi tu pewna analogia do np. fortepianu: nikt, kto nie ma słuchu muzycznego lub ewentualnie zdolności kompozytorskich, nie będzie umiał zagrać na fortepianie. Dzięki omawianemu programowi możemy tworzyć muzykę lub efekty dźwiękowe do własnych programów. Uniwersalny COMPILER pozwala umieścić wynikową wersję muzyki w dowolnym miejscu pamięci, co wielu może zachęcić do wykorzystywania muzyczek napisanych na SOUND TRACKER-ze we własnych programach demonstracyjnych. ST tworzy jedynie muzykę, którą można odtwarzać przez układ AY, lecz w żadnym wypadku nie tworzy gotowych „dem".

Czy TURBO PASCAL 3.0 dostępny dla Spectrum i stacji FDD 3000 jest kompatybilny z programem o tej samej nazwie działającym na IBM?

Radosław Gut, Lublin

TURBO PASCAL jest uniwersalnym językiem programowania wyższego rzędu. Wersja 3.0 dla Spectrum jest (prawie) identyczna z wersją dla IBM'a — różnice dotyczą grafiki i dźwięku. Oczywiście format zapisu na dysku nie może być taki sam w obu przypadkach, ale to już wynika z własności komputerów. Wersja dla Spectrum jest wzbogacona o dodatkowe procedury graficzne i systemowe, które dostępne są w naszej redakcji w formie bibliotek.

Posiadam Timex'a, FDD 3000 i monitor Neptun (...) Kiedy stawiam monitor na stacji, to przestaje ona prawidłowo działać. Co jest tego przyczyną?

Mariusz Piętka, Katowice

Stacja dysków FDD 3000 została tak zaprojektowana, aby na jej dość masywnej obudowie można było postawić monitor. Niestety projektanci nie przewidzieli monitorów, które zakłócają wszystko to, co znajdzie się w ich pobliżu. Takim monitorem jest Neptun. Postawienie go na stacji sprawia, iż stacja „wariuje".

Wyjścia są trzy. Pierwszym jest wymiana monitora na inny typ, drugim stawianie monitora jak najdalej od stacji i kontrolera. Trzecim wyjściem jest położenie między obudowę FDD a monitor dwumilimetrowej blachy, która sprawowałaby funkcję ekranu.

DIASTEMOS

W przyszłości będą dwa rodzaje fabryk: — fabryki zautomatyzowane i fabryki, które zbankrutowały

Howard Anderson (The Yankee Group)

To nie jest slogan. Od połowy lat siedemdziesiątych w krajach wysoko rozwiniętych zaczęto na skalę masową stosować roboty przemysłowe, których pracą sterowały komputery. To właśnie zautomatyzowane linie produkcyjne pozwoliły koncernowi FIAT skutecznie konkurować z japońskimi wytwórcami. Jednak produkcja samochodów jest niczym wobec najnowocześniejszych technologii wytwarzania układów scalonych.

W sterylnych halach prudukcyjnych, gdzie czasteczek kurzu jest mniej niż w sali operacyjnej, automatyzacja jest koniecznością.

W Instytucie Technologii Elektronowej w Warszawie zespół naukowców kierowany przez doc. dr Henryka Przewłockigo, opracował system DIASTEMOS, czyli komputerowy system wspomagania przepływu i analizy informacji o przebiegu procesu produkcyjnego na linii technologicznej, produkującej układy scalone w technologii MOS w warszawskich zakładach CEMI.

To była konieczność — wyjaśnia dr Tomasz Gutt, jeden ze współpracowników doc. Przewłockiego — *przemysł połprzewodnikowy charakteryzuje się coraz większym stopniem automatyzacji. Cykl produkcji układów trwa blisko miesiąc. W tym czasie konieczna jest nieustanna analiza setek informacji o warunkach panujących w halach produkcyjnych, sprawności urządzeń, jakości płytek krzemu. System DIASTEMOS pozwala na stałą analizę i kontrolę.*

Na jakość produkowanych struktur wpływa tak wiele czynników, że w chwili zakończenia procesu wytwarzania układów zwykle nie wiadomo, CO wpłynęło na ich parametry. Wilgotnosc powietrza, stan zapylenia, temperatura? Srebrzyste płytki są niezwykle wrażliwe, wymagają cierpliwości i uwagi.

W każdej chwili widać, co dzieje się z konkretną partią dwudziestu pięciu płytek — dodaje inż. Sergiusz Andrzejewski — *możemy także, na ile jest to możliwe, ustalić optymalne warunki dla osiągnięcia jak największego uzysku w produkcji. System przedstawia wyniki w postaci wykresów.*

Całością steruje centralny komputer, terminale zaś rozstawione są w halach i laboratoriach. Tam też wprowadza się dane. Razem z inż. Andrzejewskim znajdujemy się w hali produkcyjnej. To pierwsza od bardzo długiego czasu wizyta dziennikarza w tym miejscu. Przechodzimy trzy śluzy i ubrani w specjalne kombinezony wchodzimy do jasno oświetlonego pomieszczenia. Po kilku minutach czuję niezwykłą suchość w gardle.

Proszę się nie przejmować — mówi inż. Andrzejewski — *wilgotność sięga tu zaledwie 40 procent, właściwie wszyscy chorujemy na gardło. Sterylne warunki są równie przykre jak brud.*

W sąsiedniej hali w specjalnych piecach zachodzi dyfuzja różnych związków chemicznych w krzemie. To jeden z najbardziej skomplikowanych etapów produkcji. Monokryształ zmienia się w układ bramek i ścieżek, po których w przyszłości pobiegną elektrony.

Sprawny scentralizowany obieg informacji wykorzystywanych w produkcji, powstających w trakcie procesu wytwarzania, zapewniają jedynie systemy komputerowe. Podobne do DIASTEMOSA — jak COMETS firmy Consilum lub system PROMIS — najczęściej nie są sprzedawane, lecz dzierżawione i to za sumy sięgające setek tysięcy dolarów rocznie.

Oprogramowanie systemowe jest niezwykle kosztowne. Zespół doc. Przewłockiego wykonał ogromną pracę, której znaczenia dla gospodarki *nie sposób przecenić*. Pieniądze wydane na elektronikę i informatykę prawie zawsze przynoszą zysk. Jeśli nie dziś, to w przyszłosci doświadczenie naukowców zaowocuje doskonalszymi rozwiązaniami.

Przykład krajów Dalekiego Wschodu, które dzięki elektronice dokonały w ciągu ostatnich dwudziestu lat szokującego skoku cywilizacyjnego i technologicznego powinien skłonić do zastanowienia.

Wiem, że w niedalekiej przyszłości w zautomatyzowanych fabrykach praca ludzka zostanie ograniczona do niezbędnego minimum. System — jak zapewnili mnie jego autorzy — może znaleźć zastosowanie wszędzie tam, gdzie w procesie produkcji zachodzi konieczność analizy ogromnej ilości danych, np. w zakładach chemicznych. Zachodni specjaliści wysoko ocenili jego walory, a zwłaszcza „elastyczność" rozwiązań. Kilka lat temu część DIASTEMOSA została zakupiona przez Instytut Mikroelektroniki w Sofii. Bułgarzy, którzy są znacznie bardziej od nas zaawansowani w produkcji układów scalonych uznali, że to, co proponują Polacy, jest najlepsze.

Autorzy systemu DIASTEMOS twierdzą, że doświadczenia zdobyte w trakcie pracy nad nim pozwalają im na przygotowanie oprogramowania dla innych gałęzi przemysłu. Te ambitne zamierzenia są godne wsparcia.

Komputerowe wspomaganie produkcji jest codziennością w nowoczesnych zakładach. Zgodnie z maksymą Howarda Andersona przetrwają tylko fabryki zautomatyzowane. Jestem przekonany, że w niedalekiej przyszłosci wiele rodzimych przedsiębiorstw zacznie rozglądać się za zakupem nowoczesnych technologii na Zachodzie. Warto wiedzieć, że w kraju także mamy ciekawe rozwiązania.

Marek Czarkowski

Przykład krajów Dalekiego Wschodu, które dzięki elektronice dokonały szokującego skoku cywilizacyjnego, powinien skłonić do zastanowienia.