1-2'90 ЯНУАРИ-ФЕВРУАРИ ГОДИНА ШЕСТА ЦЕНА 1,60 ЛВ. ISSN–0205–189

КОМПЮТЪР ЗА ВАС

НОВИТЕ ВИРУСИ В БЪЛГАРИЯ

МИКРОТЕКСТ II

ЕПЪЛ ЕКСПО

ΛΕΩΝΙΔΑΣ

Сканиране и обработка:

Антон Оруш

www.sandacite.net
deltichko@abv.bg
0896 625 803



Комбинат Роботрон, ГДР



Графичен таблет К 6405.20

Графичният таблет (дигитайзер) К 6405 е високопроизводителен и много удобен за въвеждане на графична информация в компютъра уред – координати на точки, линии, графични структури. Работата с него е максимално облекчена благодарение на добре организираното командно меню.

Таблетът ползва серийния интерфейс RS 232, поради което работи с всички IBM PC съвместими компютри и поддържа най-разпространените програмни пакети за автоматично проектиране. С помощта на ключета уредът се настройва за работа с различни програмни продукти – разделителна способност, формат и скорост на предаване на информацията към компютъра.

Размерите на таблета са 320 × 210 mm.

За демонстрация, получаване на подробна информация относно качествата, цената и доставката се обръщайте към

Търговско-политически отдел
на Посолство на ГДР в НРБ
Секция Роботрон
1113 София
ул. Жолио – Кюри 26
Тел. 66-03-81, вътр. 551

Драги читатели,

Отърколи се старата година със своя толкова обнадеждаващ за бъдещето на всички ни завършек и навлязохме в навярно най-решаващата, поне от края на Втората световна война насам, за съдбините на България година.

Бихрушката на събитията помете много от непоклатимите довчера догми, с нетърпение очакваме и с останалите да се случи същото. От десетилетната си летаргия се отърси и периодичният ни печат, вестниците започнаха да стават кой от кой по-интересни. Да, сега вече може да се говори за журналистика и журналисти.

"Компютър за вас" обаче е списание, при това специализирано и с безнадеждно продължителен за отразяване на актуални събития цикъл на отпечатване. Така че и занапред главната ни задача и участие в процесите на преустройството ще бъде да подпомагаме израстването на умни и порядъчни, на способни и компетентни млади хора. Независимо от някои крайни мнения в протичащата напоследък дискусия, едва ли някой се съмнява, че бъдещето принадлежи на електрониката и компютърната техника. Друг е въпросът за приоритетите и цената на нейното развитие. При отчайващата техническа и технологична изостаналост в елементната ни база, поне аз не си представям, че в обозримото бъдеще ще можем да смаем света с хардуерни постижения. Шансът ни е в разработката на оригинален софтуер, за което се иска преди всичко сиво вещество. Ние безспорно разполагаме с талантливи програмисти, друг е въпросът в каква среда и при какви условия ги отглеждаме.

Съвсем доскоро хората ги деляха най-вече по два основни признака: наши и ненаши. Може и да си тъпичък, стига да си „наш", всичко ти се прощаваше. С този подход всеки знае докъде стигна икономиката ни и деградацията на моралните ценности в обществото. Време е нещата да се поставят на местата им и делението да става на кадърен и компетентен и съответно — некадърен и некомпетентен. Именно първите трябва да станат „нашите" и те да заемат полагащите им се в социалната стълбица места. А то е длъжно да ги опазва от сдружената мафия на посредствеността и полуинтелигенцията.

Крайно време е да намерят своето поне средноевропейско (избягвам вече поизлъскания жаргон „цивилизовано") решение проблемите за заплащането и защитата на труда на програмистите, развитието на частната инициатива и т.н. При това под заплащане разбирам заплащане и в „действителни" пари, т.е. в конвертируема валута, когато съответният продукт се продава за долари.

Начало на темата поставяме със статията на А. Николов за правната защита на програмните продукти. Тя е писана преди 10 ноември и затова очакваме още по-злободневни статии по тези проблеми, статии, които ще поместваме с предимство.

В броя ще намерите и поредната статия за вредностите при работа с компютър. Постоянните читатели на списанието очевидно са схванали, че съзнателно публикувахме поредица от такива материали, съблюдавайки народната поговорка: „Думам ти, щерко, сещай се, снахо". Темата не е монопол на „Компютър за вас" — и в други издания се появиха публикации по нея, тя присъстваше и на последната международна конференция „Децата в информационния век". Общо взето всичко написано и казано бе в добре познатия ни успокояващо-приспивен тон. Всички автори говореха все за компютъра и нито един не спомена, че всъщност главният вредител е видеомониторът. Един измерваше електромагнитни полета, друг — рентгеновото излъчване, но никой нема куража да каже, че в завод „Аналитик", Михайловград се произвеждат едни зеленикавосветещи мониторчета, изплъзнали се от всякакви стандарти и нямащи аналог по безобразното качество на нефокусирания трепкащ образ.

Докато набера кураж на глас да произнеса: „царят е гол", дойде новината, че от началото на тази година производството на тези монитори вече е спряно.

Поздравявам ръководството на ДФ „Микропроцесорни системи" за това решение!

Ето и обещаната допълнителна информация за домашния компютър Правец-8С. На стр.57 е посочено какво точно включва стандартната конфигурация, с която се продава компютърът. Монтирането на интегралните схеми за серийния интерфейс ще се прави като допълнителна услуга във всички поделения на ИП „Системинженеринг".

Ние поставихме пред ръководството на Комбината по микропроцесорна техника и друг въпрос, с който мнозина читатели се обърнаха към редакцията: как да се сдобият с толкова хваления (при това напълно заслужено) на страниците на списанието интегриран продукт за този компютър ПАРИС. Понеже програмната къща „Правец-Програма" разполага само с 800 бройки от него (две дискети и описание от 225 страници), тези комплекти ще бъдат включени безплатно към пуснатите в продажба в началото на тази година компютри.

За абонатите на списанието и ние правим поредното дарение, като ще им записваме безплатно ПАРИС. Така, както постъпваме с ДОС-8Д и всички по-дълги програми, чиито листинги не е възможно да публикуваме при нищожния сегашен обем на списанието.

Засега все още не сме в състояние да увеличим обема или печатните приложения към списанието и затова развиваме, при това безплатно, такива нестандартни, но несъмнено полезни и желани от читателите ни услуги.

Инж. Георги Балански

Когато в статията си в КВ.7 – 8.89 предсказах появата на нови компютърни вируси у нас в края на октомври 1989 г., не подозирах, че хипотезата ми ще се окаже до такава степен вярна. И наистина – именно през това време, за по-малко от три седмици, у нас се появиха три нови вируса. При това наблюдава се рязко покачване не само на количеството вируси, но и на тяхното качество... Създателите им започнаха да стават по-хитри, а премахването на техните произведения – все по-трудна задача.

НОВИТЕ ВИРУСИ В БЪЛГАРИЯ

Н. с. инж. ВЕСЕЛИН БОНЧЕВ

Вирусът V651

Първата лястовичка лойле, както можеше да се очаква, от един от многобройните клубове „Компютър". Случаят, за който искам да ви разкажа, се появи в образа на дискета, пълна със заразени файлове, получена от втора и трета ръка, като този, който ми я продаде лично, се кълнеше, че става дума за нов и изключително опасен вирус, който можел да заразява компютрите дори и при изпълнение на командата DIR върху заразената дискета. Тъй като последното е очевидно невъзможно, пък и потребителят, който ми съобщи това, не беше от най-квалифицираните, смело се залових за работа.

В началото (при бегло прележдане на заразените програми) останах с впечатлението, че се отнася за файлове, не особено компетентно почистени от вируса Eddie. Още повече че заедно със заразената дискета до мен достигна и слухът, че „новият" вирус е дело именно на създателя на Eddie.

При по-внимателно разглеждане излязоха наяве следните факти. Първо, наистина ставаше дума за нов вирус. Второ, създателят му явно се е блазнел от популярността на Eddie, защото, поне външно, вирусът му прилича на него. По-детайлното изучаване на вируса обаче показа, че става дума само за външни прилики. (Между другото, за мен това е сигурен признак, че и този вирус е създаден у нас.) Оригиналният Eddie е много по-сложен и очевидно е създаден от друг човек. Просто „почеркът на програмиране" е различен. В краен случай бих могъл да се заблудя, ако вирусите се бяха появили в нашата страна в обратен ред – новият би могъл с известно пресилване да бъде взет за начална версия на Eddie. Той е много по-прост от него и в известен смисъл може да се разглежда като учебен пример за това, как се пише резидентен файлов вирус. Но нека да преминем към кратко описание на самия вирус.

Той е дълъг само 651 байта (откъдето произтича и името му). Независимо от малката си дължина вирусът се изхитрява да върши повечето от нещата, характерни за къде-къде по-големите му събратя.

Преди всичко, той спада към класа на т. нар. резидентни вируси. Това означава, че при първото стартиране на заразена с него програма вирусът остава резидентен в паметта на компютъра и след това заразява всеки стартиран файл. Естествено, въпреки че е резидентен, вирусът не може да бъде „видян" с такива елементарни средства, като програмата MAPMEM.COM. За да го видите, необходимо е да използвате функцията system Info на програмата PCTools. Ако вирусът присъства в паметта, това може да бъде установено по разликата от 1 Кбайт между цялата памет на системата според PC – DOS и според PCTools. За съжаление понякога тази разлика може да се дължи и на апаратна повреда.

Вирусът заразява както .COM-, така и .EXE-файлове. Както вече споменах по-горе, заразяването на файловете се извършва при стартирането им (а не при всяка файлова операция, както е при вируса Eddie). За да бъде заразен, файлът трябва да има размери между 651 и 64372 байта.

В самия вирус (почти в самия му край) се намира низът „Eddie lives". Само толкова, без останалата част от придобилата печална известност фраза. И тук тя както и при вируса Eddie не се използва за нищо. Още една прилика между тези два вируса – променената информация от нападнатите файлове се пази в последните 11 байта от кода на вируса, при това по еднакъв начин и за двата вируса. С това обаче приликите между тях се изчерпват.

Подобно на вируса VHP – 648 (наричан понякога по света „Виенският вирус") V651 използва в качеството на маркер за заразените от него файлове полето за секунди във времето за последна промяна на файла.

Както и при VHP – 648, то се установява равно на 62. Това позволява лесно да се разработи програма, която да ваксинира изпълнимите файлове срещу този вирус. Осмелявам се да предположа, че това качество на вируса ще способства за бързото му изтребване, така както стана с VHP – 648 – сега у нас той почти не се среща вече. Още повече че тези файлове, които са вече ваксинирани срещу VHP – 648, ще бъдат неуязвими и срещу новия вирус. Просто не трябва да забравяме, че и .EXE-файловете ще се нуждаят от ваксиниране.

Вирусът V651 притежава собствена програма за обработка на критичните грешки, така че съобщението

Abort, Retry, Ignore?

да не се появява например при опит за заразяване на файл, намиращ се върху механично защитена срещу запис дискета. Но може би е излишно да отделям особено внимание на този въпрос – наличието на такава програма (critical error handler) стана едва ли не стандарт за всеки уважаващ себе си компютърен вирус. Де да беше така и с обикновените програми... Но да не се отклоняваме.

Този вирус притежава една особеност, която не съм срещал досега у неговите събратя. Особеност, която му помага да остане по-дълго време незабелязан. Тя се състои в следното.

Тъй като вирусът е дълъг 651 байта, логично е, че всеки заразен от него файл ще увеличи дължината си именно с толкова. Ако обаче вирусът е в паметта и изпълните командата DIR, ще видите... оригиналните (непроменени) дължини на файловете.

„Фокусът" се състои в това, че вирусът прехваща функциите на PC – DOS за търсене на файловете FindFirst и FindNext (метод FCB). Именно тези функции използва командата DIR, за да извлече имената на намиращите се в директорията файлове. Освен имената на файловете обаче тези две функции връщат и цял куп друга полезна информация. Между другото – времето на последна промяна на файловете и тяхната дължина. Е добре, вирусът използва първото, за да определи дали съответният файл е заразен и ако това е така, изважда 651 от числото, показващо дължината на файла.

За щастие тази иначе оригинална идея не е доведена до логическия си завършек. Така например не се прехващат функциите на PC – DOS FindFirst и FindNext, изпълнявани по метода ФМ. Като следствие от това при използване на повечето от известните програми за работа с файлове (например FILES.COM) се виждат увеличените дължини на заразените файлове. Освен това именно по метода ФМ работят повечето от програмите за ранна диагностика на вирусна инфекция, които запомнят оригиналните дължини на изпълнимите файлове. Наистина това препятствие лесно може да бъде преодоляно, като се прехванат и тези функции. Затова по-сигурни са защитните програми, които не само запомнят дължините, но и пресмятат контролни суми на защитаваните файлове. Въпреки че, както вървят нещата, никак не бих се изненадал, ако един ден и тази защита бъде преодоляна.

Освен това възможно е в директорията да се намира файл, чиято дължина да е по-малка от 651 байта и който е бил вече ваксиниран срещу вируса VHP – 648. Тогава след „корекцията", която вирусът прави с информацията за дължината му, последната ще се извежда в листинга на директорията като едно огромно число.

И накрая най-важното. Вирусът V651 е напълно безвреден. Нещо повече – той не се проявява по никакъв начин. Никакви повреди по файловата система, никакви видеоефекти, никакво свирене на популярни мелодийки. Нищо. Неговата единствена цел е да живее, да се разпространява и да се радва на живота. Въпреки това, разбира се, по добре е да го премахнете от диска си, ако той случайно вече се е настанил там. За целта съм разработил съответната програма (тя има и ваксиниращи функции), която е включена към антивирусната дискета, която е на разположение на читателите в редакцията на „Компютър за вас".

Ако описаният по-горе вирус без съмнение е създаден в нашата страна, то вторият, който се появи през описвания период от време, е отдавна известен на Запад и аз съм просто изненадан, че той се появи у нас едва сега. Става дума за една мутация на вируса „Падащите букви", или по-точно за

вируса V1704

По своето действие той е напълно аналогичен на вируса V1701, който отдавна е познат у нас и който беше подробно описан в статията ми в КВ.7 – 8.89. Разбира се, той има поне една разлика – това, че е с три байта по-дълъг. За съжаление това е достатъчно, за да не могат да го откриват програмите, предназначени за борба с по-ранния щам.

Обаче, както вече споменах, този вирус е добре известен по света. За него съществуват огромно количество лекуващи програми. Аз разполагах с някои от тях доста преди да се появи самият вирус. Но, както казват французите, най-добре е, когато човек се обслужи сам (on n'est jamais si bien servi que par soi-meme). Затова, вместо да използвам чужди програми, предпочетох да променя собствената си, така че сега тя лекува и двете мутации на вируса – V1701 и V1704.

Източникът на проникване на този вирус у нас не ми е известен, но за първи път проявата му беше регистрирана отново в един от клубовете „Компютър", и то, разбира се – закачен за една игра.

Вирусът V512

Ако мястото на създаването на предишните два вируса (в смисъл – у нас или в чужбина) не буди съмнение, то това съвсем не е така с вируса, за който се готвя да ви разкажа.

Той се появи по един доста загадъчен начин на един от компютрите в Института по математика и досега (ноември 1989 г.) нямам сведения за появата му другаде.

В самия вирус няма никакъв подпис, който да помогне при определянето на произхода му. Впрочем в самия край на вируса се намира низът „666". Става дума именно за низ, а не за число. За съжаление обаче на мен той нищо не ми говори. Във всеки случай, доколкото ми е известно, на Запад като че ли няма сведения за такъв вирус, а аз доста внимателно следя съобщенията, посветени на тези въпроси.

Вирусът е дълъг само половин Кбайт, но в тези 512 байта се съдържат толкова много и толкова дълбоки хватки, които, струва ми

се, са по силите само на български програмист. Програмист, посветил изцяло времето си на изучаването на тънкостите на операционната система (очевидно поради липса на други занимания) и познаващ я по-добре даже от самите й създатели.

За наш общ късмет вирусът попаднал на изключително квалифициран потребител – Васил Николов, студент по информатика. Докато работел на компютъра си, изведнъж му направило впечатление, че шрифтът на екрана не е този, с който е свикнал. И тъй като използвал видеоконтролер тип EGA, налагал се очевидният извод, че програмата, зараждаща знаков генератор във видеоконтролера, не е сработила поради някаква причина. Тъй като бил предпазлив, В. Николов изолирал тази програма и внимателно я реасемблирал с помощта на системната програма DEBUG.COM. Оказало се, че първите 512 байта от програмата изглеждали доста необичайно. В. Николов правилно предположил, че става дума за нов, неизвестен досега вирус и побързал да се свърже с мен. Когато се срещнахме, той вече беше успял да изолира вируса и даже да начертае приблизителна блок-схема на алгоритъма, по който се изпълняват отделните части от кода му. (Подобна квалификация прави чест на всеки потребител.) В. Николов не беше направил само две неща – не беше посмял да прави експерименти с разпространението на вируса върху компютъра, на който работи, и не беше успял да разбере какво точно прави вирусът. Първото от тези две неща е лесно обяснимо, а колкото до второто, няма нищо чудно – вирусът се оказа извънредно коварен. Аз загубих цяла седмица в напрегнато проучване на тези прословути 512 байта код, като и досега не съм убеден, че съм разбрал всичките хитрини на звяра. А на мое разположение бяха не само разнообразни ръководства (както книжни, така и електронни, като например DOS Thech Help!, The Norton Guides и др.), но и един файл, свален преди по-малко от месец от една западна компютърна мрежа и съдържащ последни сведения за употребата на всички прекъсвания в компютъра IBM PC/XT/AT/PS–2, както и за всякакви недокумснтирани „хватки". Но дори и там не са описани нещата, които създателят на вируса несъмнено великолепно е познавал. Впрочем съдете сами.

Най-напред взех следните предпазни мерки. Стартирах програмата ANTI4US.EXE. Тази антивирусна програма следи за изпълнението на някои опасни функции на PC–DOS, като запис в изпълним файл, директен запис върху диска и др. Тя е твърде досадна, за да бъде използвана във всекидневната работа, но при изучаването на нов вирус е просто незаменим инструмент. Освен това на компютъра ми постоянно работи програмата CLOCKKEY.EXE, която показва часовник на 26-ия ред, инсталира драйвер за кирилица и позволява да бъдат заключвани за запис всички твърди дискове чрез прехващането на INT 13h. Излишно е да споменавам, че преди да стартирам вируса, включих забраната за запис върху дисковете.

Ефектът от всички тези защити беше... нулев. Вирусът безпрепятствено зарази стартирания файл-капан. При повторния опит – копиране на файл – вирусът зарази както копието, така и оригинала. Също така безпрепятствено. Впоследствие се оказа, че междувременно е успял да зарази и командния ми интерпретатор (файла COMMAND.COM). Ако в този момент беше пожелал да унищожи информацията върху диска ми, можеше да го направи и аз щях да съм безсилен да му попреча.

Досега са ми известни няколко начина за работа с дисковете, които остават незабележими за следящите програми.

Единият е директното управление на дисковия контролер през входно-изходните портове. Този начин обаче е толкова трудоемък и непреносим (поради голямото разнообразие от дискови контролери), че вероятността да бъде използван от вирус, който при това е дълъг само 512 байта, според мен е равна на нула.

Вторият начин се състои в това, че вместо INT 13h се използва директно програмата в BIOS-а на компютъра, която обработва INT 13h. Наистина повечето адреси на програмите, обработващи прекъсвания, не са на фиксирано място – в смисъл че могат да се менят от един компютър на друг. Обаче една от ранните версии на PC–DOS вместо инструкцията INT 13h използва именно директно обръщение на този адрес. Поради тази именно причина почти всички производители на BIOS-програми се стремят да запазят този адрес, за да постигнат по-голяма съвместимост. Обикновено там се намира не самата програма за обработка на прекъсването, а инструкция за безусловен преход към нея. Има обаче една съществена причина, поради която този метод не се използва от вирусите. Работата се състои в това, че на този адрес се намира програмата за работа с флопидисковите устройства. Ако програмата за самопроверка на компютъра при включване (Power-On Self Test – POST) открие наличие на контролер за твърд диск с намираща се на него програма в PROM, тя премества обръщението към адрес F000:EC59 от INT 13h на INT 40h, а на INT 13h поставя обръщение към PROM-а на контролера. А както е добре известно, компютърните вируси изпитват много по-голям апетит към дисковете. Затова и достъпът, ограничен само до флопидисковите устройства, не е в състояние да ги съблазни особено.

Третият метод за заобикаляне на следящите резидентни програми се използва от вируса Eddie. Той претърсва адресното пространство, отделено за контролерите, докато намери програмата, управляваща твърдия диск – точно така, както го прави програмата за POST.

Четвъртият известен ми метод се използва от някои версии на вируса TP. Той обаче е доста сложен, за да бъде използван в един 512-байтов вирус.

Методът, използван от вируса V512, е невероятно прост и същевременно, не е описан в нито един от източниците, с които разполагам. Той се състои в извикването на INT 2Fh, като в регистъра AH стои числото 13h.

За да разбера какво прави тази функция на операционната система, наложи се да реасемблирам една нейна част. И досега не съм сигурен с каква цел е реализирана тази функция, но факт е, че след изпълнението ѝ в регистри DS:DX се получава оригиналният адрес на програмата за работа с дисковете. Едва ли са много хората, които смятат, че PC–DOS е особено user-friendly (дружелюбна към потребителя). Наличието на тази функция обаче недвусмислено показва, че тази операционна система е проектирана като virus-friendly (дружелюбна към вирусите).

Впрочем описаната по-горе функция е реализирана само в PC–DOS, версия 3.30 и по-голяма – нещо, което вирусът много добре знае и внимателно проверява. Така че при по-малки версии на операционната система резидентните защити все още ще работят.

Само описаният дотук и използван от вируса трик е достатъчен, за да илюстрира дълбочината на познанията, с които е разполагал създателят му. Фокусите обаче не свършват с това. Ето още няколко от тях.

Вирусът е резидентен. Наистина да се надявате да го видите в паметта чрез програмата MAPMEM (при знанията, с които е разполагал създателят му) е просто смешно. Каква беше изненадата ми обаче, когато даже и програмата PCTools – едно от най-мощните средства за



откриване на резидентни вируси, не открих нищо подозрително – размерът на паметта, определен и по двата различни начина, беше точно толкова, колкото трябва – 640 Кбайта.

След внимателното проучване на вируса се оказа, че... той наистина не изпитва нужда да задели памет, в която да се настани. За тази цел той използва вече заделената от PC–DOS памет за буфери. Вирусът се настанява в първия от тях, след което го изключва от списъка буфери, като насочва указателя към първия буфер на операционната система – бъдещия втори буфер. При това отново се използва една недокументирана и сравнително слабо позната функция.

Вирусът заразява само .COM-файлове. Ако трябва да бъдем съвсем точни, вирусът заразява файлове, чието разширение започва с „CO". При това ги заразява както при изпълнението, така и при копирането им. Още първото стартиране на заразена с вируса програма, предизвиква и заразяването на файла COMMAND.COM. Това се извършва чрез същия механизъм, както и при вируса Eddie – разрушава се транзитната част на командния интерпретатор, което от своя страна предизвиква неговото презареждане. Само че вирусът вече е прехванал съответната функция за работа с файлове...

Както вече неколкократно споменавах, вирусът е дълъг точно 512 байта. При това логично е размерът на заразяваните файлове да се увеличава именно с толкова. Да, ама не! Заразените файлове не променят нито размера си, нито атрибутите си, нито времето на последна промяна. Къде тогава се записва вирусът ли? Ами просто в първите 512 байта от файла. Но тогава програмите ще престават да работят! – ще възкликне наивният читател. Във всеки случай точно така възкликнах аз, когато за първи път разбрах за тази подробност. Уви, отново авторът на вируса ме беше изпреварил с една стъпка в разсъжденията си.

Какво беше учудването ми, когато (само за да докажа колко съм прав) стартирах една заразена програма (при резидентен в паметта вирус) и тя... заработи. Случайност, помислих си, просто вирусът не е засегнал жизненоважни за тази програма места.

Следващият експеримент беше проведен значително по-внимателно. Направих програма, чиито съществени функции (извеждане на съобщението „Hello world!" на екрана) се намираха именно в началото, а останалата част представляваше просто пълнеж. Заразих я, след което изключих компютъра (за да премахна резидентния в паметта вирус), включих го отново и стартирах заразената програма. Нищо не се случи, просто отново се появи промптът на операционната система.

Ето че имам право, помислих си, и (за всеки случай) стартирах програмата отново. Какво мислите, че стана? Точно така, програмата сработи!

Тъй като не вярвам в духове (нито в малките, зелени човечета, за които казват, че уж живеели в компютрите), заех се внимателно да разгледам кода на вируса, както и заразените файлове. В природата нищо не се губи и нищо не се появява отникъде, помислих си. Щом като замазаната от вируса част работи, значи тя все пак се премества някъде и след това се взема оттам. Така, въоръжен с принципите на диалектическия материализъм (и с малко здрав разум) все пак открих, в какво се състои работата.

Оказа се, че вирусът премества замазаната част... след края на файла. Както е добре известно, всеки файл заема цял брой клъстери върху диска. Тъй като обаче, в общия случай размерът на файла не е кратен на размера на клъстера, в последния клъстер обикновено остава доста свободно място. Именно това място използва вирусът.

Разбира се, далеч не при всички файлове това място е по-голямо или равно на 512 байта. Как постъпва вирусът с такива файлове ли? Отговорът на този напълно естествен въпрос е не по-малко естествен. Вирусът разпознава тази ситуация и не заразява такива файлове.

При стартиране на така „обработения" файл управлението, разбира се, се получава от вируса. Той се настанява в паметта, след което програмата приключва работата си, т. е. – не тръгва. При повторното й стартиране обаче вирусът разпознава, че файлът е заразен и извлича преместената част, поради което програмата вече работи.

Тази идея не е съвсем нова. Досега неколкократно ми е бил задаван въпросът, дали не е възможно създаването на вирус, който да се настанява в неизползваното пространство от последния клъстер на файла и който следователно да не увеличава дължината му. Винаги съм отговарял, че това е нецелесъобразно. И наистина командата COPY копира файловете не по клъстери, а по байтове. Следователно достатъчно е един заразен файл да бъде изкопиран и съществената част от вируса ще се изгуби. Ще остане само частта, която търси и зарежда тази съществена част от задфайловото пространство. Тъй като обаче в това пространство на новоизкопирания файл не се съдържа смислена програма, заразените и изкопирани файлове просто ще престанат да работят.

Горните забележки важат с пълна сила и за вируса V512, с едно малко изключение. Работата е там, че в изкопирания файл ще се съдържа целият вирус. Ще липсва само замазаната част от същинската програма. Затова при повторното стартиране на такъв файл програмата все пак няма да работи и компютърът най-вероятно „ще увисва". Обаче не забравяйте, че при първото стартиране на файла всичко ще завърши благополучно и вирусът ще се настани в паметта, готов да заразява и други файлове.

Това свойство на изкопираните заразени файлове води до един съществен проблем. Проблемът се състои в това, че не е възможно надеждното лекуване на заразените файлове. Ако те са оригинални, човек все още може да се надява да намери липсващите 512 байта в последния клъстер след края на файла. Ами ако става дума за копие? Тогава „излекуваната" програма просто ще престане да работи. Остава ни единствено успокоението, че тя поне ще бъде свободна от вируса.

Наистина подобен проблем ще възниква рядко. Не забравяйте, че при резидентен в паметта вирус и при копиране на файл се заразяват както копието, така и оригиналът. Проблемът ще възниква само за заразени файлове, копирани при липсващ в паметта вирус.

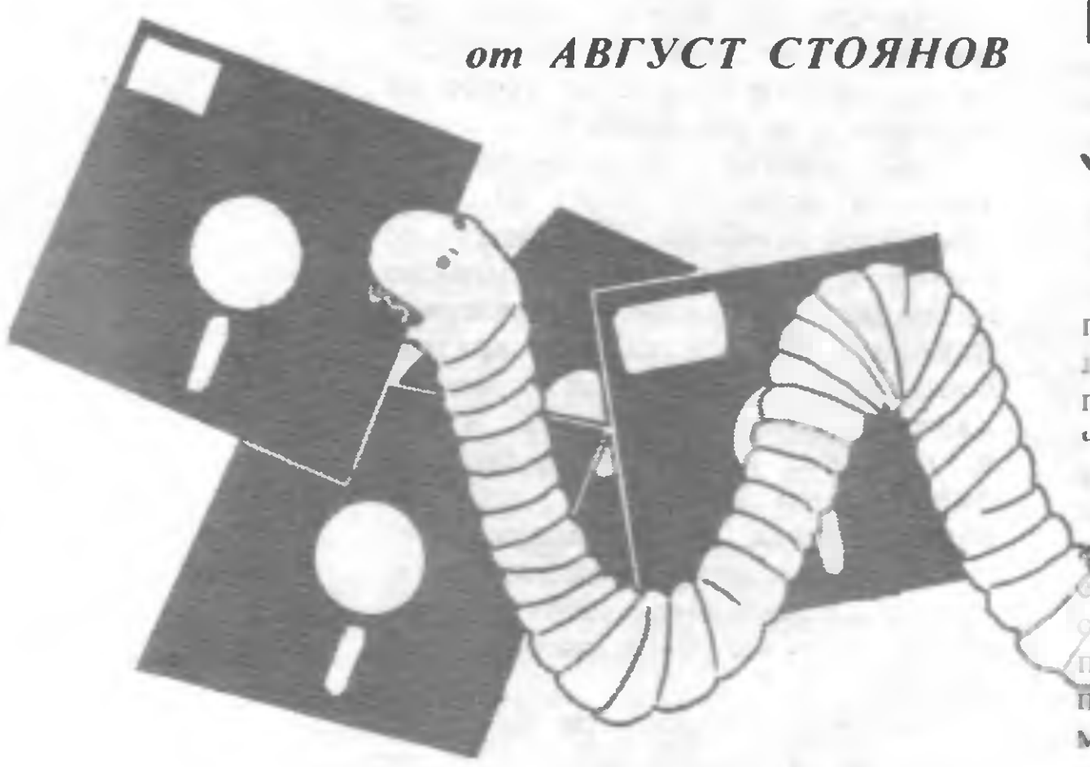
При цялото майсторство на създателя на вируса V512 той не е взел под внимание един елементарен факт. И наистина – спомнете си, че заразените програми тръгват едва след повторното им стартиране (ако в паметта още няма вирус). Спомнете си също така, че файлът COMMAND.COM се заразява първи. А сега си помислете какво ще се случи при рестартирането на системата. Тогава вирусът все още не присъства в паметта, следователно няма кой да извлече липсващата част от задфайловото пространство на командния интерпретатор. Няма и кой да го стартира повторно – операционната система извършва тази дейност, уви, само еднократно. Затова още при първия опит за рестартирането на компютъра последният „ще се забива". Този недостатък е много съществен и със сигурност ще бъде сериозна пречка за успешното разпространение на вируса.

■

ЗА ЛУДИЯ И ВИРУСНИТЕ ЗЕЛНИЦИ

1/2-философско есе за хората и зловредните програми

от АВГУСТ СТОЯНОВ



Аналогията между компютърните и биологичните вируси не е съвсем случайна – те имат обща черта, характерна за всички живи организми: необходимостта от обмен на материя и информация с външната среда. Именно това с отворената врата, която е била, е и ще бъде потенциална заплаха за разрушаването на всяка материална система. Тъй като подробното изследване на въпроса за съществуването на крайни системи, чието състояние може да се определи за произволно дълго време, е твърде обширно, ще кажа само, че колкото по-сложна е системата, толкова по-трудно е да се предвиди взаимодействието на нейните елементи помежду им, да не говорим за нейната реакция на външните влияния (стихийни и целенасочени).

Нека разгледаме сега доколко системата човек – персонален компютър е устойчива на тях (читателят ще трябва сам да направи аналогия с други системи с участието на компютри). Що се отнася до стихийните фактори като температура, влага, налягане, удар, пренапрежение, електромагнитни полета и пр., съвременните персонални компютри са на съвсем приемливо ниво (ясно е, че с безсмислено компютърът да издържа много по-страшни условия, отколкото човекът, работещ с него).

Що се отнася до целенасочените агресии, системата човек – ПК е значително по-безпомощна и перспективите в близко бъдеще не са по-розови. Тук под „агресии" имам предвид вирусите. Преобладаващото мнозинство потребители са склонни да наричат така всякакъв вид „престъпен софтуер". За да няма недоразумения, под вирус трябва да се разбира програма, която, веднъж изпълнена на даден компютър заедно с евентуално други резултати, в резултат на своето изпълнение се присъединява веднага или по-късно в оперативната или външната памет на този или друг компютър като част от същата или друга програма (в частност към операционната система), така че при последващото изпълнение на получените след присъединяването програми процесът да се повтори*). Тази процедура се нарича акт на „размножаване" или „заразяване" по аналогия с биологичните вируси и е отличителна черта за компютърните. Внимателният читател веднага разбира, че не е задължително вирусите да бъдат „убийци", разрушавайки огромни количества ценни данни, за да причинят огромни парични щети. За никому ненужното размножаване се хабят машинно време (съответно времето на хората, които чакат за него), времето на персонала, който трябва да почисти външната памет, задръстена с копия на вируси, и времето на всички онези, които се занимават (професионално или любителски) с вирусни проблеми. От дълбока древност, а още повече днес времето е скъпо във всеки смисъл на тази дума. Като вземем предвид незабележимостта и мащабите на действие на вирусите, очевидни са възможностите за използването им за „лично удоволствие", провокация и шантаж.

Много хора, склонни към драматизъм, и любители на зрелища поставят вируси, програми и компютри от едната страна на „барикадата", а от другата – беззащитните, невинни потребители, които всеки момент могат да бъдат „изправени до стената за разстрел". А истината е, че и компютрите, и програмите се създават от хора, така че за разпространяването и за унищожаването на вирусите са били, са и ще бъдат виновни хората. Тук според мен ударението трябва да се постави другаде – учудващо е колко много хора трябва да поделят тази вина. Фирмата IBM заля световния пазар с персонални компютри, чиито хардуер и операционна система (на фирмата Microsoft) представляват почти идеална среда за разпространението на вирусите. Между другото – конструкторите на първите варианти на тези компютри без съмнение са знаели, че системата им е съвсем беззащитна, но тогава този проблем не вълнувал никого, просто идеята за подобна дейност не беше назряла.

---

* **Б.Р.** – *Може би следното рекурсивно определение е по-ясно. Нека наречем присъединяваща се програма (ПП) такава програма, която, след като се изпълни, се присъединява към други програми. Тогава вирус е ПП, която превръща програмата-приемник в ПП – Славчо Иванов.*



В следващите модели софтуеристи и хардуеристи, подценявайки лошите страни на пълната съвместимост, са изложили потребителите си на риска от масово „поразяване с един удар". Архитектурата на съвременните микропроцесори, компютри и мрежи дава възможност за работата на „непробиваеми" операционни системи, но тъй като се е наложило последните да бъдат написани отчасти или изцяло наново, в тях са допуснати (случайно и/или от немарливост) грешки, представляващи потенциални „дупки" в системата, от които се възползват „бандитите".

Според мен **основната вина за разпространението на вирусите носят (особено у нас) потребителите.** И наистина — представете си персонален компютър, който не е включен в мрежа с други. Очевидно единственият начин той „да се зарази" без знанието на потребителя е да се изпълни заразена програма от дискета. Но откъде се вземат заразените програми? Не ми е известно някоя софтуерна фирма да е пуснала в продажба заразен продукт (тя няма сметка да го прави — пазарният механизъм ще я ликвидира веднага щом стане известно, че предлага зловреден продукт). Следователно заразената програма е била копирана по „пиратски" или е предоставена от някой приятел, както се прави на драго сърце помен с чужда пита. И в двата случая отговорността е изцяло на потребителя. Възможно е, разбира се, „бандитът с вирус в ръка" да зарази компютъра, докато потребителят лежи на пода, упоен с хлороформ, но такива криминални истории са извън рамките на нашето разглеждане. Всъщност **у нас вирусите са едно заслужено, макар и незаконно, наказание (поради липса на законни) за всички, които присвояват чужда интелектуална собственост, като крадат софтуер.**

Но тези разсъждения се отнасят за разпространението на зловредните програми. Какво да кажем за тези, които пишат вирусите? Следвайки народната мъдрост за зелниците, трябва да им припиша най-малка част от вината. Но в мотивировката им могат да се различат нюанси**):

• Трогателната искреност на някои от създателите на вируси може да разнежи всеки, който страда като тях от неизживяно детство и мания за величие.

• Други смятат, че по този начин „се обличат по последна мода" или избиват комплекса си за малоценност, издигайки се на „световно равнище".

• Разбира се, възможно е и някой отличен програмист, на когото не са платили както се полага, да „сложи шапка" на своите работодатели, за да си отмъсти или да ги изнудва. Затова, както е казал един много известен специалист по маркетинга, **добре е на добрите програмисти да се плаща добре.**

• Що се отнася до вирусите, разпространяващи се у нас, те са далеч от върховете на програмистото майсторство.

За всички, чийто мотив е самочувствието — съжалявам, но трябва да ги разочаровам: елементарно е да се напише вирус за MSDOS на IBM PC. Значително по-трудно е да се разпознава дали някоя програма е вирус, или не, да се отстранява или да не се допуска. Тук има поле за изява на добрите програмисти. Но те като правило нямат желание да се занимават с тази дейност. Те са най-слабо засегнати от действието на вирусите (по понятни причини) и не им се ще (и с право) да си губят времето с писане на програми, които да пазят останалите потребители.

В заключение ми се иска да обобщя: компютърните вируси извоюваха мястото си на няколко типа съвременни компютри (най-вече IBM PC) и да се пренебрегва опасността от тях е проява на престъпна небрежност. Въпреки това моят оптимизъм е съвсем основателен. Надявам се в най-скоро време всички програми, предназначени за масовия потребител на най-застрашените типове компютри, да притежават вградени защити (различни за различните програми и за различните им версии) срещу вируси. По този начин разпространението на даден вирус ще се ограничи върху минимален брой програми (на практика за всяка програма — отделен вирус), тъй като преодоляването на няколко различни защити ще направи вируса огромен и забележим с „просто око". Усъвършенстването на антивирусните програми (които **се заплащат**) ще принуждава вирусописците да губят все повече време за усъвършенстване на вирусите (които **не се заплащат**) и в крайна сметка да се откажат да ги пишат. Крайно време е и у нас да се узаконят санкции, които да ускорят този процес. Нарастващата информираност и квалификация на масовия потребител ще затруднява значително разпространението на вирусите и последствията от тях. Разумът ще надделее над пестеливостта и системи за съхранение и преносяне на ценна информация няма да се правят с използване на евтини бзащитни компютри и софтуер.

В края на краищата вярвам, че всички, имащи пряко или косвено вина за компютърните вируси, ще се принудят да споделят и отговорността, в резултат на което ще се вземат необходимите мерки за премахване на опасността.

■

** **Б. А.** — *По мои лични наблюдения и впечатления, без никакви претенции нито за строгост, нито за изчерпателност на класификациата.*

● i – борса ● i – борса ●

ИЗЧИСЛЯВАНЕ НА УСПЕХА

(регистрационен № 1.А097.02206 — 01)

Програмният продукт „Успех" е предназначен за въвеждане, съхраняване и обработка на информация, отразяваща постигнатите резултати от учебно-възпитателната работа в едно учебно заведение за период от един срок или една година. Той обаче може да бъде използван от всички видове основни и средни училища, техникуми.

Машинна конфигурация. Персонален микрокомпютър Правец–82 с 48 Кбайта оперативна памет и две дискетни устройства.

Единична цена — 40 лева.

Разработчик и разпространител:
София,
бул. „Антон Иванов" 5
НИПЛ „Програмно осигуряване"
Телефони: 627754.
62561 (557, 581)

● i – борса ● i – борса ●

К.ю.н. АСЕН НИКОЛОВ

ПРАВЕН РЕЖИМ НА ПРОГРАМНИТЕ ПРОДУКТИ

Развитието на изчислителната техника през втората половина на този век превърна програмите, чрез които работят електронноизчислителните ма- шини, във важна и много доходоносна стока. Така се появи въпросът за уреждане на правата, които възникват при създаването на програмите и при сделките с тях. Това уреждане може да стане по задължителен начин с актове, издадени от съответната държава (нормативни актове), които съдържат съответните правила (правни норми). България е първата страна в света, която издаде нормативен акт за програмните продукти – Наредба № 8 за разработване и разпространяване на програмни продукти от 1982 г. на бившия Държавен комитет за наука и технически прогрес. Когато започна да се стимулира допълнителното производство на стоки и услуги от граждани, през 1987 г. Комитетът по информатика издаде Наредба № 1 за разработване на програмни продукти и извършване на програмни услуги чрез колективната и личната дейност на граждани. Доколкото програмните продукти представляват по своето естество научноизследователски и технологични продукти, към тях се прилагат и някои разпоредби на Правилника за прилагане на Указ № 56 за стопанската дейност (ППУСД) и на Наредба за стопанските договори (НСД). Поначало така изброените актове се прилагат в отношения между различни участници в стопанската дейност. Докато ППУСД се прилага относно всички, Наредба № 8 урежда само отношения между тези, които могат да се определят като социалистически организации, т. е. между държавните фирми, общинските и кооперативните фирми, образувани от тях, както и държавни учреждения. Към същите отношения се прилага и НСД. Наредба № 1 пък се прилага в отношения между граждани, които не са регистрирали фирма, относно техните отношения със социалистически организации и относно по-нататъшното движение на програмните продукти, разработени от посочените граждани.

ПРОГРАМНИ ПРОДУКТИ И УСЛУГИ

С наименованието „програмни продукти“ в нашето законодателство се предава съдържанието на това, което на английски се определя като „софтуер“. Колкото и да изглежда странно, относно определението на софтуера съществуват редица спорове. Затова у нас нормативните актове го определят. Според Наредба № 1 програмен продукт е една или повече програми, които решават определена задача, заедно с документацията им, а програма е серия от инструкции, изразени на машинен или програмен език и записани на технически носител по начин, който позволява да бъдат използвани на електронноизчислителна техника за постигане на определен резултат.

С Наредба № 1 беше въведено още едно понятие – програмни услуги. Такива услуги са:

– **въвеждане в действие (инсталиране) на програмни продукти на определени електронноизчислителни технически средства или оказване на помощ при въвеждането им в действие;**

– **разработване на отделни програми или модули от програмни продукти;**

– **модификация на програмни продукти, което означава изменение на съществуващ програм продукт, за да се задово. конкретни потребителски потр ности;**

– изготвяне на документац на програмни продукти;

– оказване на системна пом относно програмни продукт което означава разясняване начините за решаване на задач те, поставени от потребите в процеса на работата с програ ни продукти чрез електро ноизчислителна техника;

– отстраняване на недостат и грешки в програмни продукт

– оказване на консултантс помощ при разработване програмни продукти относ спецификата на задачите, кои трябва да се решават, или съобразяване на състава и ор низацията на програмните пр дукти с потребностите на потр бителите.

С оглед да се разкрие въ можност за извършване и програмни услуги, които не били взети предвид при изготвя на наредбата, накрая е добавен още една група – „други програ мни услуги“.

РАЗРАБОТВАН НА ПРОГРАМН ПРОДУКТ

Организация или граждан могат, най-общо, да разработя програмни продукти по дв причини – заради това, че няко друг, наречен „възложител“, и възложи това и им го заплат или защото разработващия прецени, че ще има изгода о разработването и пристъпи към него без възлагане, със сво сили и средства. Наредба № 8 ус тановява определен ред, който трябва да се спазва при разработ ването. Текстовете са редакти рани с оглед на разработване при възлагане, но се прилагат и тогава, когато се разработва без възлагане, като в този случай разработващият („изпъл нителят“ по терминологията на наредбата) извършва и това, което е предвидено за възложи теля. При разработване по

възлагане задължително се сключва договор между възложителя и изпълнителя. Както вече беше посочено, Наредба № 8 се прилага в отношения между социалистически организации.

Преди започване на работата (преди да се сключи договор при възлагане), възложителят е длъжен да състави технико-икономическо задание и да го изпрати в Националния програмен и проектен фонд, който в 15-дневен срок му предава справка. В справката се посочва съществуват ли в страната еквивалентни или сходни програмни продукти, които решават изцяло или частично същия проблем, а ако е възможно — и такива чуждестранни програмни продукти. Посочват се и техните цени. Въз основа на справката и други проучвания, възложителят решава дали да започне разработването, като евентуално го възложи на изпълнител.

Разработването преминава през три етапа: изготвяне на задание за разработка (Р1), алгоритмизация и програмиране (Р2) и опитна експлоатация и внедряване (Р3). В заданието за разработка се уточнява и допълва технико-икономическото задание, определят се параметрите на програмния продукт. Въз основа на него се определя по-нататък съответствието на разработения продукт с поставената задача. За съдържанието му има български държавен стандарт. То се приема чрез утвърждаването му от възложителя. На етап Р2 се разработва опитен образец, който включва програмен код, записан на носителя, и документация с обем и вид, уточнени в заданието за разработка. Разработването се приема по начин, определен в договора. Етап Р3 включва следните работи: опитна експлотация и проверка на програмния продукт, поправяне на откритите при тази експлоатация грешки и съставяне на програмна документация. Етапът завършва с приемни изпитания или внедряване на програмния продукт по начин, определен в договора.

Изпълнителят е длъжен, след като се утвърди заданието за разработка, да изпрати копие в Националния програмен и проектен фонд за получаване на регистрационен номер, а в двумесечен срок от приключване на етап Р3 да изпрати пак там анотация на програмния продукт. Ако пък разработването бъде преустановено преди приключването му, изпълнителят е длъжен в 14-дневен срок да уведоми за това фонда.

Според чл. 4 на Указ № 56 на всички фирми се осигуряват равностойни условия за осъществяване на стопанска дейност. Затова, макар че Наредба № 8 има предвид само отношения между социалистически организации, очертаният ред ще се приложи и при програмни продукти, които се разработват от граждани. Ако обаче програмни продукти се разработват от граждани, които не са регистрирали фирма, редът е друг. Такива програмни продукти трябва да се предложат за изкупуване според Наредба № 1 на социалистическа организация. Регистрацията им се извършва от организацията, която е изкупила програмния продукт, т. е. след етап Р1. При такива продукти не съществува изискване за предварително искане на справка, нито са задължителни всички етапи, предвидени от Наредба № 8, но при всяко положение трябва да съществува задание за разработка.

ПРАВА ВЪРХУ ПРОГРАМНИТЕ ПРОДУКТИ

Според чл. 15 на ППУСД научноизследователските и технологичните продукти, които не са обект на особена правна закрила, принадлежат на:

— **фирмата, в която са създадени;**

— **възложителя, когато са създадени в изпълнение на договор;**

— **гражданите, които са ги създали самостоятелно извън изпълнението на служебни или договорни задължения.**

Програмните продукти са именно от тази категория. Наредба № 8 говори, че стопанин на програмния продукт е възложителят в смисъл, че продуктът му принадлежи. Наредбата не урежда изрично правата върху програмни продукти, създадени без възлагане, защото изхожда от предпоставката, че в тези случаи изпълнителят е същевременно и възложител (възложител сам на себе си).

РАЗПРОСТРАНЕНИЕ НА ПРОГРАМНИ ПРОДУКТИ

Както вече беше посочено, сега програмните продукти са важна и твърде доходна стока. Етапът, при който те се разпространяват у нас безплатно, беше приключен с излизането на Наредба № 8. Сега в огромния брой от случаите те се разпространяват срещу заплащане, т. е. те се продават.

Продажбата на програмните продукти обаче съдържа редица особености в сравнение с обикновената продажба. Програмният продукт представлява ценност не като съвкупност от вещи (хартия и технически носители), а с интелектуалното съдържание, което е фиксирано върху тях. По тази причина продуктът може да се копира с минимални разходи и да се разпространява в неограничен брой екземпляри. Пак по тази причина, докато обикновена вещ (например машина) може да се притежава и ползва само от един, програмен продукт може да се ползва от всеки, който притежава копие от него.

Наредба № 8 различава притежаването на програмния продукт в неговата цялост (правата върху интелектуалното съдържание) от притежаването на продадено копие. Притежаването от първата категория, както беше посочено, е на възложителя, съответно — на фирмата, в която е създаден продуктът, или на гражданите, които са го създали извън изпълнение на служебни или договорни задължения. Възложителят според Наредба № 8 може да разпространява продукта (да продава копия), като го отстъпва за ползване на потребители, т. е. на такива, които ще го използват в своята дейност. Потребителят може да използва продаденото му копие само за нуждите на собствената си организация — той няма право да го продава или да го предоставя безплатно. Наредбата предвижда и друга възможност за отстъпване на ползването — на определена ЕИМ. В тези случаи продуктът може да се използва

от неограничен брой организации, но само на определена в договора ЕИМ. Правата си на разпространение възложителят може да осъществява сам или чрез организация, на която той е предоставил разпространението. Както при отстъпване на ползването (продажба на копия) на потребители, така и при предоставяне на друга организация, е необходимо да се сключат съответни договори.

Посоченият чл. 15 от ППУСД създава по-големи възможности за притежателя на програмния продукт. Според него фирмите и гражданите с договор могат да отстъпват или да предоставят за използване от други фирми продуктите, които им принадлежат. Под „отстъпване" следва да се разбира възможността да се прехвърли притежаването в неговата цялост. По отношение на програмни продукти на граждани, които не са регистрирали фирма, Наредба № 1 съдава твърде различен режим. Такива програмни продукти могат да се разпространяват само от социалистическата организация, която ги е изкупила. Под „изкупуване" се разбира прехвърлянето на социалистическата организация на притежаването в неговата цялост. Освен това такива програмни продукти могат да се разпространяват само ако са придружени с документ за съответствието им със стандартизационните и другите установени изисквания. Според заповед № РД 07 – 1/1988 г. на председателя на бившия Комитет по информатика документът се издава, след като се провери съответства ли продуктът на задачата, поставена в заданието за разработка, както и спазени ли са изискванията на БДС или отраслови нормали при изготвяне на документацията. По искане на организацията, която е изкупила или желае да изкупи програмния продукт, проверката се извършва от следните организации:

– **Предприятие за програмни продукти „Микроинвест" – за програмни продукти за автоматизация на проектирането при инвестиционната дейност;**

– **Централен институт по изчислителна техника и технологии – за операционни системи, транслатори, компилатори, базови програмни продукти за системи за телеобработка и сервизни програми, включително комплект програми за техническо обслужване;**

– **Институт за техническа кибернетика и роботика при БАН – за базови програмни продукти за осигуряване на работа в мрежа на ЕИМ;**

– **Комбинат „Национален програмен и проектен фонд" – за всички останали програмни продукти с изключение на такива за автоматизация на технологичните процеси.**

МНЕНИЕ НА НАУЧНИЯ КОНСУЛТАНТ

Още при публикуването на Наредба № 1 бях изненадан от непълното определение на програмен продукт — едно от най-важните понятия на компютърната информатика.

В наредбата **програмен продукт** се определя като **една или няколко взаимосвързани програми**, записани на външен носител и съпроводени с **документация**.

Известно е, че всяка машинна програма използва известен брой константи — числови и знакови, които се разполагат в данновите полета на програмата. Обикновено те се намират плътно зад блока на машинните инструкции. Но има програми (програмни системи), които използват огромен брой предварително събрани и съхранени данни (константи), които в никакъв случай не се разполагат в тялото на програмата и се намират на външен носител. Например разпространява се програмен продукт, който представлява готова база от данни по география или история и пр., заедно със СУБД, която управлява тази база.

Ето защо в определението на понятието програмен продукт е наложително да намери място и **трети** компонент освен програми и документация, който не е задължителен — и, **евентуално, данни извън програмата, записани на външен носител.**

Не е необходимо да се споменава, че всеки държавен документ трябва да използва точно и **пълно** определение на обектите, с които се занимава. Отговорност за посочения пропуск носят единствено информатиците, които са били консултанти при изготвянето на цитирания документ.

Искам да отбележа и нещо по-важно — това **не е** определение на програмен продукт, а определение на понятието **програмно осигуряване (ПО)**.
А **програмен продукт** е **програмно осигуряване** със следните **три свойства**:

1. **ПО е написано професионално.** Това не означава — от професионалисти, понеже има професионалисти, които са типични дилетанти.

2. **ПО е предназначено да се използва по принцип от външни лица — потребители.** Това не означава, че и авторът (авторите) му не могат да го използват.

3. **ПО е изготвено, за да се продава като търговска стока.** Възможно е обаче един програмен продукт да не се продаде нито веднъж, защото е остарял, неудобен, документацията му е лошо написана, не е ефективен или, най-сетне, е много скъп.

Доц. ДИМИТЪР П. ШИШКОВ

Продължение от КВ, 11 – 12.89

ЕПЪЛ ЕКСПО

УТРЕШНИЯТ ДЕН НА ПЕРСОНАЛНИТЕ ПРОДУКТИ

ДИМИТЪР ВАВОВ



На втория ден от конференцията, която протече в рамките на „Епъл Експо'89", пленарен доклад под заглавието „Трансформации" изнесе Майкъл Спиндлер – президент на „Епъл Европа". Той разгледа теоретично въпроса за промените, които настъпват в управлението днес. Отчитайки нарастващия риск и несигурност в бизнеса, в противоположност от предсказваното развитие до 80-те години, Спиндлер в серия от графики наблегна върху човека като основен ресурс на информационното общество

Днес целта на фирма като „Епъл" е да постигат не повече с повече капитали и хора, а много повече с много по-малко хора. Вместо за фабричния работник вече се говори за „работник на знанието". За служителите на новия тип компании са важни три ключови момента: мотивация, обучение и продуктивност.

Статистиката в САЩ сочи, че бързонарастващата производителност в страната като цяло е резултат основно от автоматизация на производствените процеси. Приносът на автоматизираната канцелария е не- значителен, защото по принцип тя преповтаря бюрократичните структури и не е в състояние гъвкаво да реагира на динамична среда.

В противоположност са фирми като „Епъл" с голямото предимство на своята младост. Те не носят наследство на традиции, централизирани бази от данни, многочислена работна сила, която трябва да се преквалифицира. Според Спиндлер досегашният термин АСУ – автоматизирана система за управление, трябва да промени съдържанието си. Автоматизираната информация не служи вече само на управлението на фирмата – на мениджърите. Тя е за всеки. Никой днес не желае да играе ролята на „обикновен терминал". Новата структура тип „мрежа" се различава принципно от традиционната бюрократична пирамида. В мрежата се търси такава гъвкавост, която да позволи събирането на най-добрите специалисти по решаване на даден проблем и разпускането им след приключване на задачата. Йерархичните нива са минимум, свободата на действие в името на творчески цели е максимум.

Необходими са прости системи – лесни за експлоатация. Сега доминиращ фактор е потребителят, а не централният компютър в автоматизираната система. Отчитайки наличността на централизирани системи на обща стойност над 500 млрд. долара, „Епъл" апелира информацията в тях да стане достъпна за масовия потребител. Тези системи не могат да се бракуват, но те трябва да се отворят. Големите компютри все повече трябва да играят ролята на сървери, обслужващи произволен персонален компютър.

Спиндлер също демонстрира част от света на Макинтош. Демонстрира по много убедителен начин в края на 1989 г. това, което „Епъл" нарече преди четири години „Мак офис" (Мак канцелария). След включването си Мак IIci даде главно меню, изпъстрено с функционално безценни картинки. Две от тях са за работа в MS – DOS среда и VAX среда – резултат от упоритото настояване на Джон Скъли, че Макинтош трябва да се свърже със средата на IBM и DEC, след като всички сериозни фирми години наред са се снабдявали с такава изчислителна техника. Конфликтът Скъли – Джобс в голяма степен бе породен от очевидното техническо превъзходство на Макинтош. Навремето това подхранваше у Джобс илюзията, че „Епъл" може да живее като голяма фирма, изолирана в своя свят –

чудесен за професионалните специалисти, но икономически безсмислен за бизнесмените в големите фирми.

На останалите картинки на екрана на Мак има всичко, което талантливият чиновник може да желае. Интегрирани продукти, всевъзможни помощни средства и главно – издателска система, с която исторически „Епъл“ се наложи като доставчик за фирми, а не само за образованието и университетите. Писането и оформянето на доклади е изведено на ниво професионален издател. Естествено в най-скъпия си вид освен лазерния принтер системата може да включва и цветен скенер.

Задачата, която реши Спиндлер, бе тривиална – доклад до началника – в случая по продажбите на портативния Макинтош. Това, което демонстрира, обаче включваше приемане на данни за очакваните продажби от MS–DOS среда – таблица на Лотус 1–2–3 и данни за реализираните продажби от мрежа на DEC, работеща в реално време. И двете таблици бяха преработени в многозадачен режим – скачайки незабележимо от продукт в продукт (сменяйки едновременно и операционните системи) в тримерни графики. В края на краищата всичко беше изсипано в издателската система, където стоеше текст и цветно изображение на самия портативен компютър – въведено с цветен скенер.

Смазваща бе лекотата на работа с графичния интерфейс на Макинтош спрямо това, което за всички нас не буди особени съмнения – командите на MS–DOS. Досадно изглеждаше логическото именуване на устройствата a: b: c: и т. н., изписването на директории и имена. При Мак просто се работеше с картинка на флопидиск или на твърд диск. Влизането и излизането от продукти и операционни системи ставаше толкова безконфликтно, както изборът на файлове от една директория в ДОС. Демонстрираната от „Епъл“ мишка през 1983 г. при Лиза днес е добила абсолютен професионализъм в графичния свят на Макинтош.

Самият канцеларски софтуер (продукти като 4-th Dimension и WingZ) работи хармонично и елегантно, предлага привлекателни допълнения. Така изборът на част от тримерната графика, създадена с 4-th Dimension, извеждаше в прозорче на екрана цифровите данни, стоящи зад графиката.

Спиндлер трябваше да покаже принципната разлика между работата с Макинтош и работата със съвместим с IBM компютър. Към това трябва да се прибави и видяното по щандовете на експото. Оценката на резултата е потискаща – „Епъл“ се намира наистина на качествено друго ниво...

На третия ден от Епъл Експо пленарен доклад на тема „Промени“ изнесе Алан Кей – мечтателят ясновидец. Изглеждащ далеч по-млад за своите 49 години, Алан Кей е човекът, за когото се казва, че е измислил почти всичко хубаво в света на персоналните компютри: от идеята за персонален компютър, през тази за портативен компютър, към тази за говорещия „навигатор на знания“ – компютъра на бъдещето. От праграмната среда с прозорци през езика Лого към графичната среда и релационната, търсеща допир с изкуствения интелект, среда на Хиперкард.

Основна част от творческата му работа през 60-те и 70-те години протича в „Ксерокс“ – в прочутата лаборатория PARC (Palo Alto Research Centre) в Силициевата долина. В една уникално свободна творчески за времето си среда, скипът на Кей моделира бъдещето на интелигентните работни станции.

„Обичам да работя върху дългосрочни проекти – усмихва се Кей, – с поне малко романтика в тях. Не ми говорете за маркетинг и производство, говорете ми за идеи.“

Още през 1968 г. той рисува DynaBook – „динамичната книга“ – всъщност задание за портативните компютри. Дайнабук не е базирана на технологии – това е просто верен поглед в бъдещето. Приликата със сегашните преносими (lap top) компютри е поразителна.

Първата работна станция е проектирана в Пало Алто от скипа на Ксерокс през 1973 година. Първата издателска система – пак там през 1977 година.

Твърди се, че в лабораторията на Кей от 1975 г. е можело да се видят основни качества на компютрите от края на 80-те... Кей и колегите му не само дадоха термина „personal computer“ – те положиха основите на това, което по-късно се прие последователно от „Епъл“ и IBM – лек за работа екран с меню от малки рисунки, наречени „икони“, управляван с малко устройство – мишка.

Преди 20 години лабораторията на Алан Кей е била пълна с прототипни разработки на компютри. Днес тя е пълна с деца – тайната за успеха на мечтателя Кей.

Критичното просветление на Кей е наистина преди 20 години. Тогава той разбира, че компютърът не е просто средство, сечиво по пътя на познанието. Той е комуникационен носител, основен като книгата. Обикновената грамотност означава способност да се чете и пише. Според Кей компютърната грамотност е не само ползване на компютъра (четенето), но и програмирането му – аналогия с писането. „И както е при молива и хартията, това не е носител, ако децата, не могат да го ползват.“ За да стане програмирането леко, е необходим лек и ясен език – такива са символните езици. Крайният тест за Кей стават децата – те именно му помагат да формулира точни задания към компютрите на бъдещето. Кей е убеден, че човек не разбира същността на нещата, които прави автоматично. Точно такъв е случаят с комуникирането. За да се извлекат верни принципи при комуникация с компютрите, трябва да се обърнем към децата. Когато са достатъчно малки, те правят всичко естествено.

Основната работа на Кей става разработката на принципно нов интерфейс за работа с компютъра, за компютърен език, който да направи революция в програмирането – да го направи достъпно за всеки. По този път се появява графичният интерфейс, езикът Лого, после СмолТок, после ХиперТок, за да се стигне днес до средата ХиперКард, от която се очаква революционната промяна.

Следва

Двама млади научни работници от Българската медицинска академия изучават въпросите на зрителното възприятие от различна гледна точка – единият от позициите на офталмологията, а другият – от позицията на хигиената на труда. Те излагат мнението си за влиянието на работата с дисплей върху зрението на оператора. Авторите разполагат с обширна литературна справка по проблема, която те предоставят чрез редакцията на списанието на интересуващите се читатели.

- Отблясъците на екрана
- Невидимите врагове: рентгеново и електромагнитно лъчение, електростатично поле
- Очила за операторите
- Задължителен годишен преглед от офталмолог

Обратната страна на медала ПАЗЕТЕ ОЧИТЕ ОТ ДИСПЛЕЯ

Д-р РАДУИЛ ЦЕКОВ
Д-р ВЕРИСЛАВ СТАНЧЕВ

Научно-техническата революция постави пред съвременната медицина редица проблеми, породени от общуването на човека с машините. Типичен пример е масовото навлизане на персоналните компютри във всички сфери на труда и бита през последните години. На тази плоскост възникват извънредно много нови проблеми, сред които един от най-важните е вредното влияние на дисплея върху зрението. С материала, посветен на рисковите фактори, свързани с работата пред дисплея, продължаваме нашата серия от публикации за обратната страна на медала компютризация (виж КВ.03.88, КВ. 9–10.88, КВ.5–6.89 и КВ. 9–10.89).

Без да навлизаме в технически подробности, следва да отбележим, че като цяло произвежданите в нашата страна монохромни и цветни дисплеи не отговарят напълно на съвременните изисквания.

Един от най-важните им недостатъци е гладката повърхност на екрана, отразяваща срещулежащи предмети или странично падаща светлина, с което се влошават зрителните условия за работа на оператора.

Добър пример в това отношение са част от дисплеите на фирмата „Videoton“ (Унгария), които са с матиран екран, или някои от дисплеите на фирмата „Robotron“ (ГДР) с прикрепено към екрана синтетично фолио. И в двата случая ефектът е намаляване на отблясъците и отстраняване на отраженията от околните предмети върху екрана.

Както бе споменато в КВ. 11.86, създадени са и специални решетки за поставяне върху екрана, които освен другите предимства премахват и статичното електричество. Впечатленията ни от употребата на подобна решетка на фирмата „Vidamic“ (Швеция) са много добри. Материалите от двата световни конгреса на тема „Работа с видеотерминал“ – Стокхолм 86 и Монреал 89, показват, че рентгеновото излъчване може да се пренебрегне като вреден фактор на работното място, тъй като интензитетът му е далеч под съществуващите норми.

Излъчваните от дисплея електромагнитни лъчения от нейонизиращия спектър са в широк честотен диапазон. Изследванията показват, че на 50–70 cm от екрана интензитетите на радиочестотните и микровълновите полета са в границите на съществуващите норми.

Що се отнася до електростатичното поле, то негов източник не е мекото рентгеново излъчване (КВ.5–6.89), а високият потенциал на ускоряващия електрод. Електростатичният заряд се увеличава и от зареждането на екрана от попадащия върху него електронен сноп.

Интензитетът на електростатичното поле често може да е на границите на нормата или дори да я надвишава.

Той може да се намали, като се повиши влажността на въздуха чрез подбор на подови настилки и облекла на операторите с минимално съдържание на изкуствени материали и др.

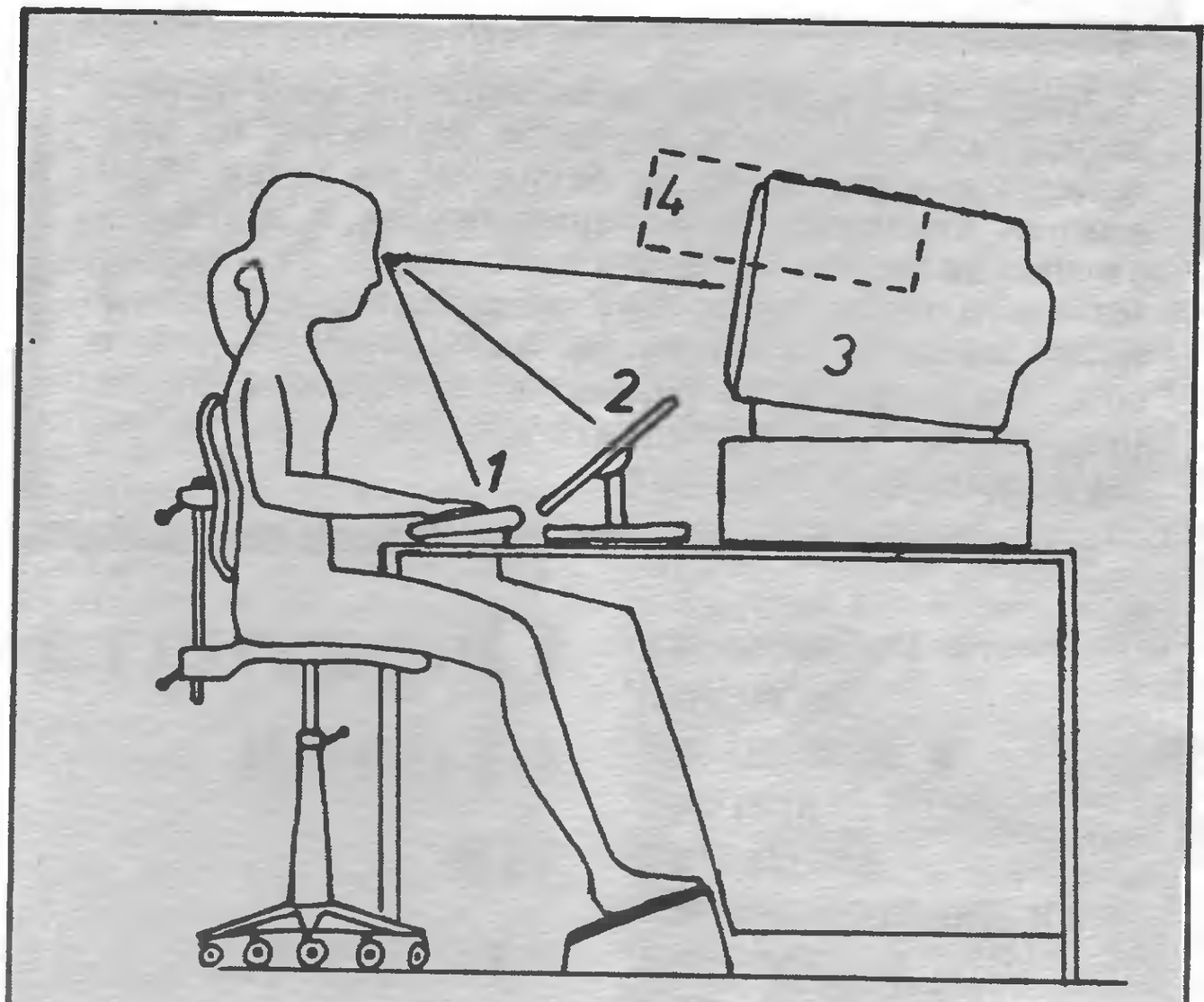
От друга страна, основно значение за продължителна работа на зрителната система има характерът на извършваната от оператора дейност. Работата с дисплей може да се раздели условно на три типа режими – диалогов, въвеждане на данни и следене на процес.

И трите имат специфични особености, но и една обща черта – те са активни процеси, което ги отличава съществено по характер на очните движения и по условия на околната среда от гледането на телевизия например. При това диалоговият режим е най-разпространен. Той може да включва разнообразни дейности – текстообработка, съставяне или работа с електронни таблици, бази от данни, използване на изчислителни или управленски програми и др., изискващи да се следят с погледа различни обекти – екран, клавиатура, данни (най-често текст).

Оттук произтичат и основните изисквания на ергономията. Преди всичко работната мебел трябва да бъде с регулируеми параметри, за да се пригажда към всеки потребител. Освен това трите зони, които се следят от погледа, трябва да се намират на приблизително еднакво разстояние от очите (фиг. 1), за да се намали умората от непрекъснатата промяна на акомодацията.

Това може да се осъществи например, като текстовете се поставят на специална стойка, приблизително на същото разстояние от очите, както екрана на дисплея.

Що се отнася до осветеността на работното място, засега в света все още няма единно мнение каква норма да се препоръча. Затова у нас



ФИГ. 1
Примерно подреждане на работно място с видеотерминал

1 — клавиатура
2 — стойка за текст
3 — монитор
4 — сенник за монитора

се използват препоръките, дадени във „Временни указания за проектиране на работни места с видеотерминал", разработени от д-р В. Койчева (Институт по хигиена при БМА). В тях се изисква общото осветление на работното място да бъде от 300 до 500 лукса. Добре би било осветеността върху самия дисплей да е по-ниска и върху него да не попада пряка светлина. Това може лесно да се постигне с поставянето на сенник. Друго важно условие е в стаята да няма големи отразяващи повърхности (бели и гладки стени, стъклени или прозрачни пластмасови плотове, големи огледала и т. н.). За правилната работа на зрителната система е необходимо също прозорецът да не попада в зрителното поле на оператора. Препоръчва се върху прозореца да има поставени щори с вертикални ламели. Както и оптимално осветление, защото най-важното условие за успешна работа е състоянието на зрителната система.

Веднага следва да отбележим, че обикновено състоянието на зрението на оператора (най-вече неговата зрителна острота) е по-лошо, отколкото смята той самият. Не повече от 35% от възрастното население на страната има аметропична, т.е. „идеална" рефракция. От изследване, проведено в ЦНИКА – София, нашият опит показва, че близо 40% от служителите, работещи с дисплей, имат нужда от корекция на зрението си (да започнат да носят очила или да подменят наличните). Друго изследване в ГДР сочи, че около 30% от работещите с дисплей носят неподходящи очила. Както е известно, нарушенията на зрителната острота се делят основно на три групи – хиперметропия (далекогледство), миопия (късогледство) и астигматизъм.

Освен тях значение за зре-



нето имат отслабването с възрастта на акомодацията (пресбиопията) и функционалното изключване на едното око от двуочното зрение (афакия).

Някои от горепосочените нарушения могат и да се комбинират помежду си.

Тук няма да описваме подробно как протичат тези аномалии, само бихме искали да отбележим някои характерни оплаквания при липсата или неправилна корекция на зрението.

Такива са например:

- гложде́не, често мигане, сълзене, зачервяване на очите
- замъгляване, размазване и сливане на буквите
- удължаването на буквите в някаква посока (при астигматизъм)
- болки в слепоочията, тежест и натиск в очните ябълки, челно главоболие.

Съвременните средства за корекция на тези аномалии включват предписването на обикновени и специални видове очила. Тук ще бъдат разгледани накратко само специалните: бифокалните, трифокалните и вариофокалните очила. От тях засега само бифокалните се продават и монтират в България.

Предимството на трифокалните очила е в монтираната преходна зона между далечния и близкия фокус.

Вариофокалните въобще нямат линия на разделяне между фокусите за близо и далеч, като оптическата им сила плавно се променя във вертикална посока.

Друга група очила са тези с филтриращи лещи, които „отсичат" от спектъра определена дължина на вълната или имат променлив коефициент на поглъщане на светлината.

Тук спадат видовете „Uro", „Umbral", „Rose" на фирмата „Zeiss" (ГДР) и прословутите „Heliomatic" на същата фирма, които се произвеждат и в България под името „Фотозал". Последните бяха рекламирани и в нашия печат като особено подходящи (в. Вечерни новини, бр. 117/89 г., стр. 2) и масово препоръчвани от оптиците в софийската градска оптична мрежа като универсално средство при работа с компютър.

По този повод заслужава да се отбележат два момента.

Първо, **не съществува универсално средство за повишаване на зрителната работоспособност при работа с дисплея.** Това се постига чрез комплекс от мероприятия, по-важните от които бяха споменати по-горе.

Второ, **времето за достигане на максимума при потъмняване или просветляване на фотохромните стъкла, каквито са „Фотозал", е сравнително много по-голямо от времето за свиване или разпускане на зеницата.**

Следователно тяхното прилагане има смисъл най-вече при висок интензитет на осветлението на работното място, което рядко се среща. Някои фирми са разработили лещи за очила с многослойно покритие против отблясъци („Antireflex" на „Zeis" (ГДР) и фирми от САЩ и Франция), но те също се препоръчват при висок общ интензитет на осветлението.

Съществуват и специално разработени за работа с компютри очила с различно по интензитет специално зелено покритие в различните зони на лещите (фирмата „Essilor" – Франция).

Според проспекта на фирмата те значително подобряват контрастността на изображението.

Цената на най-простия модел от тези очила обаче е около 40 долара. Когато се прави избор между зрителна корекция с очила и с контактни лещи, необходима е индивидуална преценка от лекар – офтамолог, макар че в преобладаващото мнозинство от случаите да е ясно, че за предпочитане са очилата.

Досега няма доказана зависимост между работата с дисплей и възникването на очни заболявания като глаукома, катаракт и др. Мнението на водещите офталмолози у нас е, че компенсираната глаукома не е противопоказание за работа с дисплей или гледане на телевизия. Някои заболявания – като конюнктивити и блефарити, могат да се провокират или усложнят от наличието на статично електричество.

Това става, тъй като електростатичното поле задържа прах, бактерии и други микрочастици, които могат да доведат до алергична реакция, главно с очни симптоми. Заслужават внимание проучванията на изследователи от ФРГ относно акомодацията в покой, показват достоверни отклонения след един час работа пред дисплея.

Българските изследвания потвърждават тези резултати.

Логичният извод е, че на всеки час работа пред дисплея е наложително да се прави минипауза – от 5 до 10 минути за релаксация на очните мускули, и да се предотврати акумулирането на зрителната умора.

Ако горните условия се спазват, възможна е и многочасова работа с дисплей без настъпване на зрителна умора. В заключение трябва да подчертаем, че операторите, работещи повече от четири часа дневно пред дисплея, както и всички, които имат по-тежки аномалии в рефракцията, е необходимо да бъдат преглеждани профилактично от лекар офтамолог всяка година.

■

КАД СИСТЕМА



за проектиране на жакардови тъкани

Ръчният цикъл на проектиране на жакардови тъкани се състои от следните етапи:

- Проектиране на тъканите от художник десенатор. Резултатът от тази операция е рисунка с условни цветове, съответстващи на местата с различна структура.
- Проектиране на сплитките за всеки условен цвят.
- Проектиране на плата във вид на рисунък на сплитката на плата.
- Кодиране на сплитката на плата на перфолента за жакардов стан.

При необходимост последните един до три етапа се повтарят няколко пъти. Операциите в тях са трудоемки, монотонни и ниско производителни.

Настоящата система има за цел да се автоматизират процесите в тези трудосмки операции. При нейното построяване стремежът е бил да се запазят максимално действията и навиците на десенаторите, изработени в съществуващия технологичен цикъл.

Ето защо за входни данни в системата служи цветна рисунка в условни цветове, а като изход се получава файл за перфориране на лента за жакардов стан или за програмиране на епром.

Минималната компютърна конфигурация, на която пакетът програми може да работ
е следната:

– IBM PC/XT съвмести
компютър
– Минимална памет 2
Кбайта, препоръчвана памет 5
Кбайта
– Твърд диск с обем поне
Мбайта
– EGA адаптор
– Цветен монитор (препоръ
ва се EGA монитор)
– Специализиран скенер
– Таблет или мишка.

Специализираният скенер с
използва само на първия ета
от проектирането.

Пакетът програми се състо
от подсистема за сканиране н
изходната рисунка, подсистем
за обработка на сканирани
файл, подсистема за редактиран
на обработения файл и средств
за дефиниране и редактиране н
сплитката.

СКАНИРАНЕ

Поради високата цена на
цветните скенери и поради
невъзможността да се получи
полутоново изображение от
нормален черно-бял скенер се
използва специализиран хардуер
за сканиране на рисунката-проект
(авторско свидетелство № 60723
собственост на ДФ „ТЕСКОМ“).
Той се състои от плотер, на
който вместо писец е монтирана
оптическа глава за четене на
ивичен код. Сигналът от главата
се усилва и подава на единия
вход на компаратор, другият
вход на който е свързан с 8-битов
ЦАП, управляван от компютъра.
Тактът на един от двигателите
на плотера е свързан със сигнала
IRQ2 на компютъра. С този
елементарен хардуер е възможно
да се сканира изображение в 256
градации на сивото с разделителна способност 8/1000 инча
при скорост на главата 30 cm/s.



Програмата сканира изображение в произволен участък от масата на плотера, извършва усредняване с „плаващо средно“ (по желание на оператора), машабира резултата, показва контролно изображение на екрана и го запазва на диска.

Компресирането на изображението е съществен момент от сканирането, тъй като рисунките нерядко са с размер 3000 × 2000 точки, което при директен запис заема 6 Мбайта дискова памет. Компресирането, което се прилага в програмата, намалява този обем 2–3 пъти.

ОБРАБОТКА НА ИЗОБРАЖЕНИЕТО

При използваните бои и техники за рисуване разликата в интензивността на отделните области с еднакъв цвят е доста съществена. Това налага създаването на самостоятелна програма за разпознаване на областите с еднакъв цвят в рисунката. Ефективността на компютърната обработка при подобни задачи е слаба и обикновено не се прави на персонални компютри. Затова в дадената система се прилага итеративно определяне на областите.

При изобразяване на полутоновото изображение 256-те градации се съпоставят на 16 цвята. Съответствието градация-цвят се дефинира от потребителя и може да се променя леко. Разделянето на области се състои в настройка на цветовете, така че желаната област да има един цвят, а границите ѝ – други.

След това точка от областта се посочва с курсора. Областта може да се дефинира в който и да е от начините за представяне на рисунката.

Областите могат да се припокриват. В припокриващата се част е валидна по-късно посочената област.

Итеративното определяне на областите се оказва значително по-бързо и точно от автоматичното, особено при нископроизводителен компютър. Това важи дори за големи картини с много на брой малки области. Например една голяма за решаваната система задача се маркира от оператор с 1–2 дни стаж за 2–3 часа, докато автоматично това може да се направи за около 20 часа (на IBM XT 4.77 MHz).

РЕДАКТИРАНЕ

Третият етап от работата по проектиране на жакардовата тъкан е редактирането на областите и присвояване и редактиране на сплитките за всеки цвят.

Редакторът обработва произволни рисунки с размери до 16 000 × 16 000 точки. Рисунката се пази във вид на области. Допускат се и многосвързани области. Обемът памет, заеман от една област, е приблизително пропорционален на периметъра ѝ. Максимално една област може да заема 64 Кбайта памет, което не е практическо ограничение за системата (средният обем памет за една област е 3–5 Кбайта).

Програмата зарежда само тези области, които са нужни за функционирането ѝ в момента, което на практика изключва откази поради липса на памет.

Системата за редактиране се управлява чрез менюта, които се избират с мишка или таблет. По-често използваните функции могат да се извикват и с клавиши.

Редакторът работи с три слоя: рисунката; слой, наречен DEF1; слой, наречен DEF2.

Във всеки слой може да се намира произволна рисунка. Еднаквите области се пазят само в едно копие в паметта и на диска.

За да се изобрази повече от един слой едновременно, е възприето рисунката да се изписва на нечетните точки, а другият слой – на четните.

По този начин, когато една област се избира за модифициране, визуално тя изглежда по-ярка. Начинът на изобразяване се задава от потребителя.

Областите от слой 1 могат да се подлагат на произволно линейно преобразование, да се мултиплицират, да се оцветяват в един цвят и други.

Между областите от слой 1 и 2 могат да се извършват множествените операции обединение, сечение и разлика, като резултатът се записва в слой 1.

Произволни цветове от рисунката могат „да се защитят“, като оцветените в тях области стават недостъпни за писане.

В произволен слой от рисунката могат да се рисуват точки, полилинии, правоъгълни области и сплайни (квадратични B-сплайни), като дебелината на „писеца“, с който се пишат, се избира от потребителя. Полилиниите и сплайните могат да бъдат затворени, отворени и запълнени.

Областите от рисунката могат да се копират в слой 1 и 2, а тези от слой 1 да се копират в рисунката.

Рисунките, създадени чрез редактора, могат да се зареждат на произволно място в друга рисунка.

ФУНКЦИИ, СВЪРЗАНИ С ТЕХНОЛОГИЯТА

На всеки цвят може да бъде присвоена определена сплитка. При дефинирането на сплитките се задава размерът ѝ, който може да бъде различен за всеки цвят, и се въвежда самата сплитка. Така дефинираната сплитка може да се запази на диска за по-нататъшна употреба. Като се има предвид, че при досегашната технология един производител борави с 10–20 различни сплитки, на практика създаването на сплитките е еднократна операция.

Единични суперпозиции, с цел да се коригират границите между областите, се допускат на всяко място в рисунката.

Като резултат се получава файл с изходен формат, удобен както за перфориране на жакардова перфолента, тип „Вердол“ на автоматизирана перфорираща машина ЛП-30, така и за програмиране на епром на устройства за жакардов стан.

Системата, към която е приложен описаният пакет програми, се произвежда от ДФ „ТЕСКОМ“ под търговската марка ЛОГИТРОН 2000.СМ.01.

■

МИКРОТЕКСТ II

Инж. ГЕОРГИ БАЛАНСКИ

Още незаглъхнали коментариите и страстите, които предизвика с безспорните си качества от една страна, а от друга — със защитата си, шедьовърът на „Микросистеми“, продуктът за автоматична корекция ПРЕСТО!, насочваме вниманието на читателите си към поредната новост от същата програмна къща — МикроТЕКСТ II. Това не е проява на пристрастие от наша страна, а поне три са причините, поради които сме струва да представиме продукта:

— МикроТЕКСТ II е нова, значително по-мощна версия на и без това популярната едноименна текстообработка и без всякакво съмнение може да се заяви, че не само за момента, но и за доста време напред той ще властва на пазара на текстообработките. А едва ли има нужда да се доказва, че именно този вид програмни продукти са най-разпространените — и у нас, и в чужбина, където от години държат около 60% от количеството на продажбите.

— за автоматична корекция се използва познатият вече ПРЕСТО!;

— продуктът бе тестван най-задълбочено и заядливо в редакцията в продължение на близо три месеца.

МикроТЕКСТ II предлага всичко, което може да се желае от една професионална текстообработка. С него удобно и бързо може да се:

— пишат и редактират всевъзможни по обем и сложност текстове в един или няколко документа едновременно;

— извършва автоматична корекция на буквени грешки;

— пренасят думите в края на редовете;

— използват в максимална степен възможностите на принтера за шрифтово разнообразие и оформяне на текста;

— отпечатва текстът в няколко колони;

— въвеждат забележки под линия;

— създава автоматично азбучен указател и съдържание;

— пишат автоматично циркулярни писма и етикети с адреси;

— сортира текст;

— номерират автоматично списъци и т.н.

Накратко — с продукта се пишат, форматират и отпечатват текстове, чертаят се елементарни графики, правят се аритметични пресмятания. Възможностите на МикроТЕКСТ II са толкова много, че те не могат да бъдат обхванати отведнъж и бързо да се изучат. Към това трябва да се добави, че голяма част от командите са трикратно дублирани — могат да се изпълняват чрез клавиатурата от командните менюта или с натискане на комбинации от клавиши за бързо въвеждане на команди, а така също и с мишка.

Това обаче не означава, че работата с МикроТЕКСТ II изисква подготовка, различаваща се съществено от работата с други по-елементарни текстообработки.

За ежедневна работа и за писане на несложни текстове е необходим минимален обем от познания. А това, че продуктът притежава още много неизползвани възможности и тънкости, не е беда — те се овладяват постепенно, според възникналата потребност.

Колкото до тези, които имат опит в работа с МикроТЕКСТ/16 или ДОКС 2.0, преминаването на МикроТЕКСТ II не представлява никаква трудност.

В една статия не е възможно подробно да се опишат всички функции на продукта — ръководството за работа с него съдържа 322 страници. Ето защо накратко ще разгледаме само по-съществените новости, които го отличават от предната версия МикроТЕКСТ/16. Разчитаме и на факта, че тя (заедно с „братовчед“ си ДОКС 2.0 на „Програмни продукти и системи“) са безспорно най-разпространените и съответно най-добре познатите у нас текстообработки.

Новата версия на продукта работи с макрокоманди. Макрокомандата е съставена от няколко автоматично изпълнявани в зададена последователност команди, които иначе би трябвало да се въвеждат една по една. Така се пести немалко време и се икономисва монотонното и многократно повтаряне на едни и същи действия. Макрокомандата се обмисля и съставя вед-

ник, след което се записва в речника и може да се стартира многократно чрез въвеждане на кода й.

Макрокомандите се създават по два начина:

– чрез последователно записване на въведените с натискане на клавиши команди;

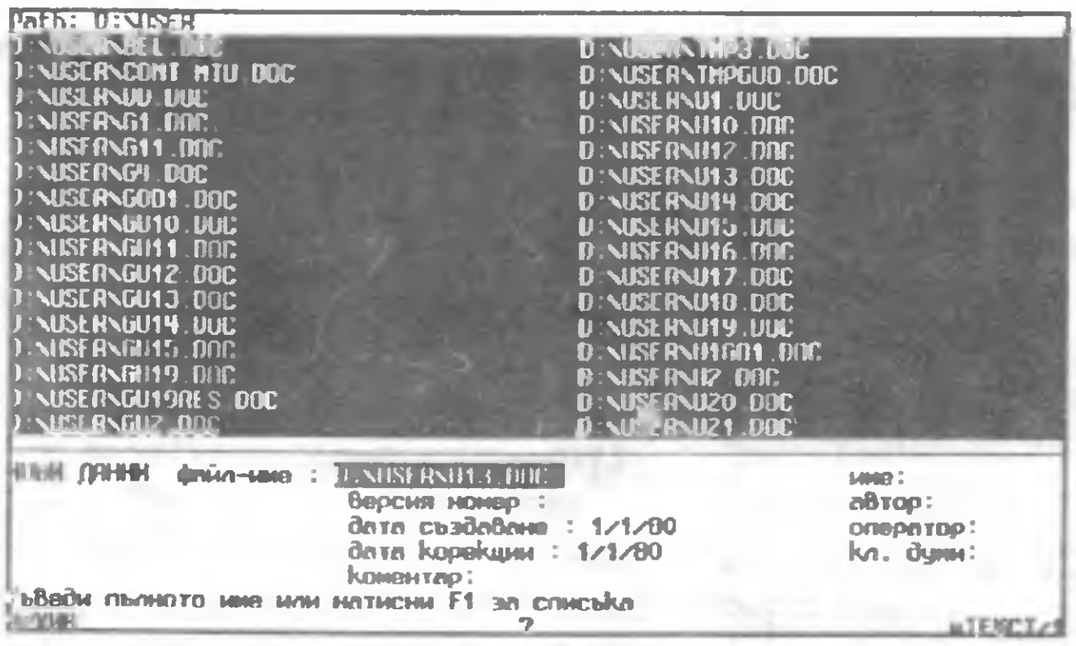
– чрез описване на поредицата команди, които трябва да се изпълнят за постигане на дадена крайна цел. В този случай в макрокомандата могат да се включат коментари, както и оператори за разклоняване на процеса на извършваните действия (чрез оператори за условен преход), логически оператори, да се въвежда допълнителна информация от клавиатурата и т.н.

В комплектацията на МикроТЕКСТ II е включен файлът MACRO.GLY, който съдържа набор от готови макрокоманди за автоматизиране на едни от най-често извършваните действия: копиране на текстове, придвижване на маркера, записване на документ или на част от него във файл, трансформиране на форматирането с маски в директно форматиране, писане на таблици, съставяне на съдържание и азбучен указател и т.н.

Имената на тези макрокоманди, кодовете за тяхното стартиране и описанието на действието им са дадени в справочника, намиращ се на трета корица на този брой на списанието.

МикроТЕКСТ II поддържа електронна картотека към записаните документи. За всеки от тях се създава бланка с информация за имената на автора и оператора, за версията на документа, за датата на създаването и на последната му редакция, за разширено заглавие на документа, ключови думи за по-лесното му намиране, дължината на текста в брой знаци.

По този начин се преодоляват трудностите при намиране на даден документ, породени от ограниченията, наложени от ДОС – имената на файловете да съдържат до осем знака.



Съдържанието на бланките може да се обновява, включително и да се попълват бланки за документи, създадени с предната версия на продукта.

Издирването и зареждането на документ чрез картотеката става изключително лесно и бързо, колкото и много файлове да има на диска.

Файловете с търсените документи могат да се търсят чрез задаване на един или повече признаци, включени в бланките.



Колкото повече са признаците, толкова повече се стеснява обхватът на документите, които ще отговарят на определените изисквания.

Даден документ може да бъде търсен по неговото име, по ключови думи, по датата на неговото създаване (или на последната му редакция), по съдържащ се в него характерен текст и т.н.

Могат да се използват и логически оператори (И, ИЛИ, НЕ, >, =, <), чрез които допълнително се стеснява кръгът от документи, които отговарят на зададените признаци. Това позволява например да се търси документ едновременно по името на автора му, в чиято бланка да се съдържа дадена ключова дума, името на оператора да е различно от посоченото и датата на създаването му да се намира в зададен интервал от време.

Новост е и възможността за включване в текста на таблици, създадени със специализирани продукти за електронни таблици – МикроПЛАН/16, МикроПЛАН II, Суперплан, Lotus 1-2-3, Multiplan, Excel. При това форматирането на таблицата практически се запазва, защото полетата от отделните колони са разделени помежду си с по един табулатор и всеки ред завършва със знак за нов ред, а не за край на абзац. За да се възстанови нормалният вид на вмъкнатата таблица, трябва само да се определи стъпката на табулаторите.

Много съществено е, че МикроТЕКСТ II съхранява информация за името на файла с електронната таблица и при промяна в нея е в състояние да актуализира копието ѝ в документа.

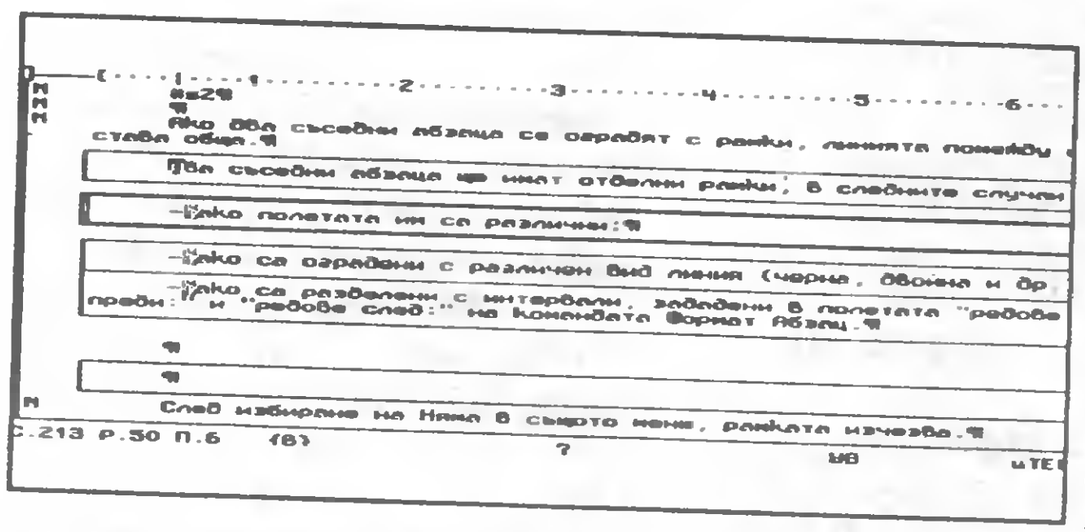
Възможно е да се вмъква както цяла таблица, така и част от нея.

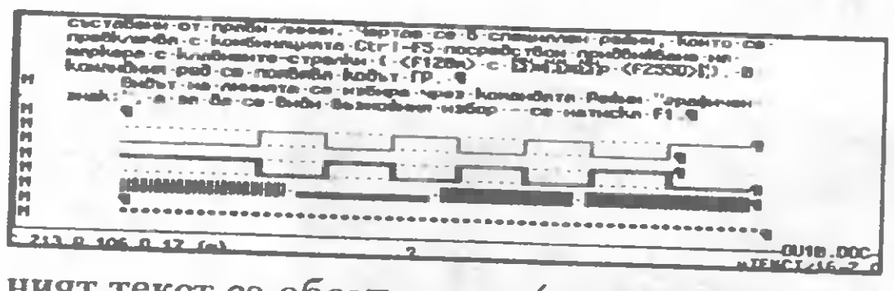
МикроТЕКСТ II предоставя и една друга полезна възможност – по време на отпечатване на даден документ на предварително определени места в него да се отпечатват и различни графични изображения, предварително създадени с подходящи графични редактори, като Chart

на Майкрософт (бизнес-графика) и др.

Освен въвеждане на текстове с продукта може да се чертаят линии около текста или той да се обгражда в каре, както и да се изчертават несложни схеми. Това се вижда добре и на илюстрацията.

МикроТЕКСТ II предоставя възможност за работа в режим, при който всички извършени корекции и нововъведе-





ният текст се обозначават (примерно с шрифт или подчертаване), за да се отличава от останалия. Изтритият текст не изчезва от екрана, а остава, но вече зачертан. Може да се трие само нововъведен текст.

Така всички извършени промени се открояват и при необходимост текстът може наново да се коригира или поправките да се анулират.

Едва след като се вземе окончателно решение, внесените промени се потвърждават и въвеждат за постоянно в текста или пък се отменят. Направените корекции могат да се търсят автоматично. МикроТЕКСТ II разполага с различни възможности за маркиране на корекциите в отпечатания, а на екрана — с вертикална черта отстрани на редовете.

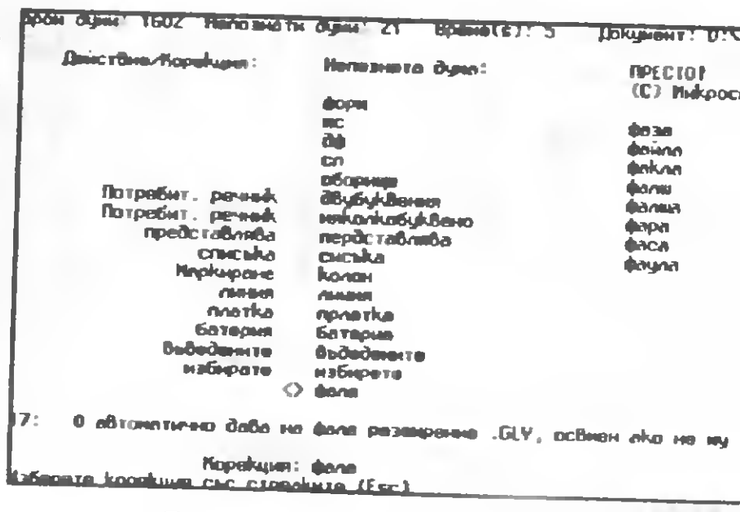
Новост е и възможността за автоматично търсене и заменяне на форматиращи текста команди, включително и на маски.

Въведено е номериране (при желание) на редовете при отпечатване на текста.

Облекчено е създаването на нови маски чрез записване на параметрите на вече форматиран текст в маска.

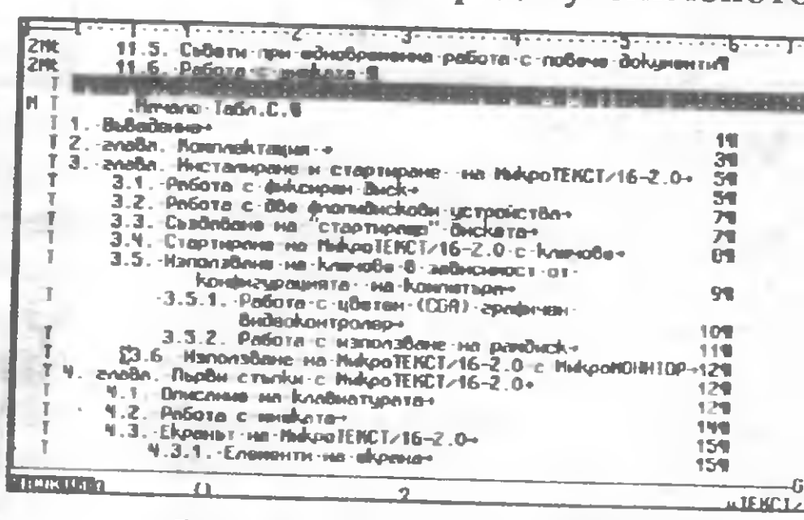
Създаването на съдържание при структурирани документи става изключително лесно (достатъчно е да се натиснат четири клавиша), възможност, която също липсва при предната версия на продукта.

В МикроТЕКСТ II е вградена най-съвършената за момента у нас програма за автоматична правописна корекция на текстове ПРЕСТО!. Тя се стартира директно от текстообработващата програма, поддържа речник от над 500 000 словоформи на около 48 000 думи от български език и извършва изключително бързо проверката — за 10 секунди около 1 000 думи (5 000 при PC AT)!



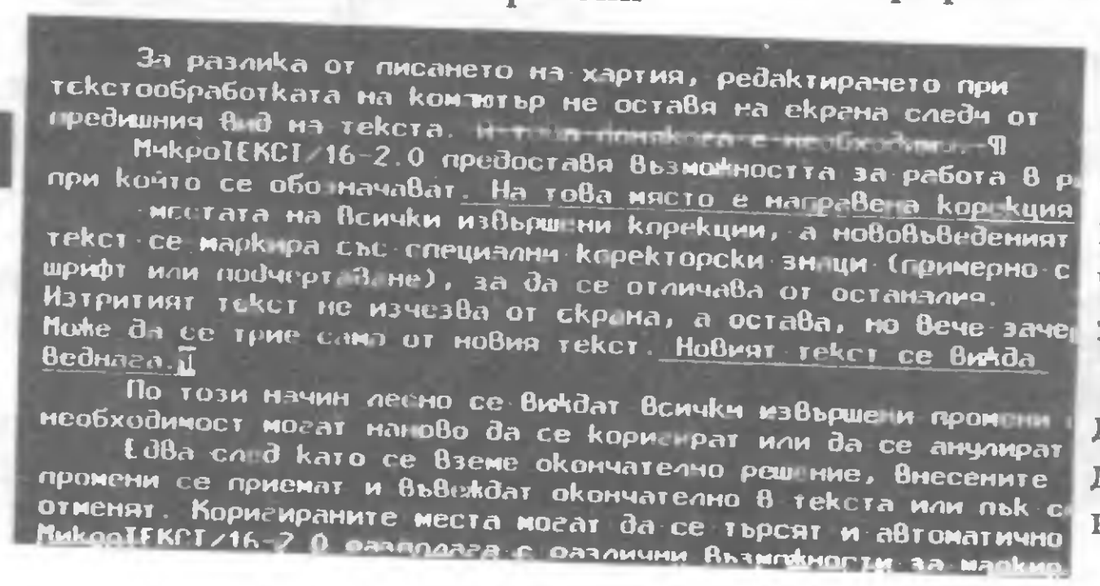
Освен основния речник ПРЕСТО! поддържа и потребителс речници, които се попълват с специфичните за обработвани текстове думи. Повече ПРЕСТО! може да прочетете „Компютър за вас“ броеве 8-9 10-11 от 1989 година.

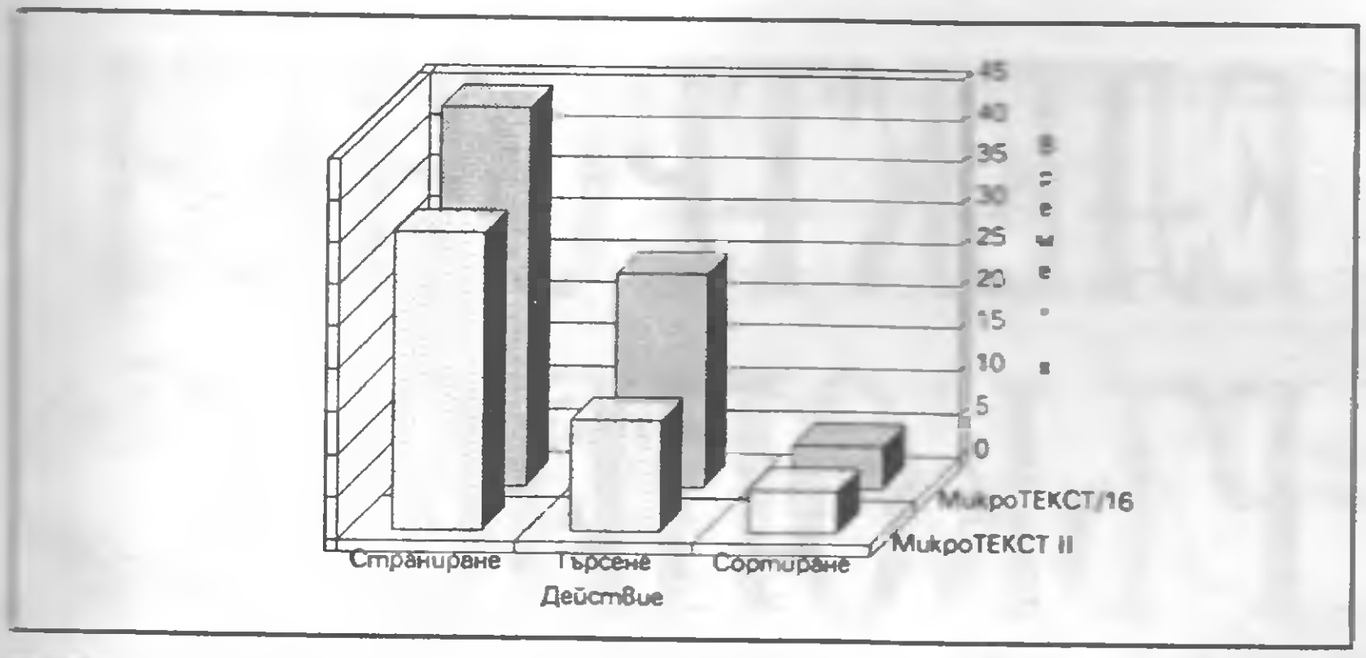
Наред с изброените дотук новости МикроТЕКСТ II притежава и **множество други подобрения.** Така например той работ значително по-бързо в сравнение с първата версия на продукта. Това се забелязва веднага – още при „прелистване“ на текста на екрана, като разликата бързината е толкова по-очевидна, колкото компютърът е по бавен. Като се има предвид, че голямата част от компютрите нас са PC XT-съвместими (тактова честота 4.77 Mhz) и особено пък, ако ви се е паднала комбинация с туткавия староза горски десет мегабайтов твърд диск, само заради увеличеното



бързодействие си заслужава преминаването на МикроТЕКСТ II.

На графиката е показано сравнение между бързодействието на двете версии, като за действията страниране и търсене е използван текст с дължина 345 000 знака. Времето за сортиране на абзаци в библиографска справка



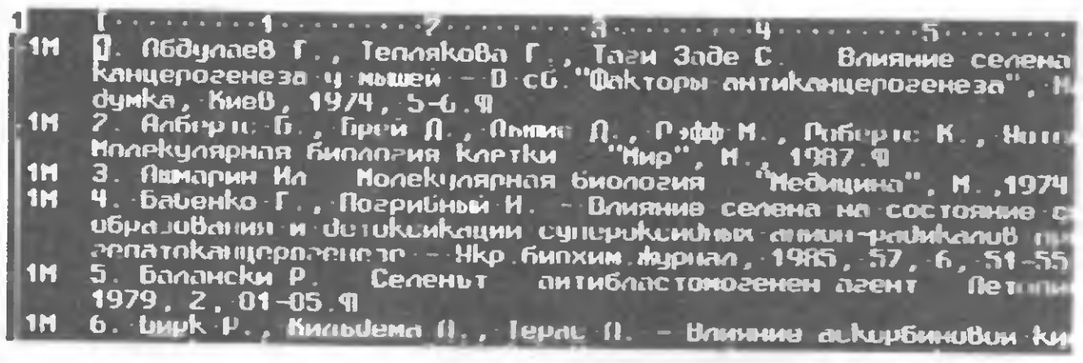


и при двете версии е еднакво, а максималната дължина на текста, с която те успяха да се справят, е около 30 000 знака или в случая – 215 абзаца.

Друго голямо удобство е възможността командите в командните менюта да се избират и чрез придвижване на маркера с клавишите-стрелки. Освен това почти всички основни и съответно най-често използвани команди могат да се въвеждат директно чрез комбинация на функционален с някой от управляващите клавиши – Ctrl, Shift или Alt. Таблицата с тези команди бе публикувана в справочника „Компютър за вас“ в КВ.11-12.89.

Освен посочените новости много от наследените функции са допълнени и разширени – за избиране на части от текста, за променяне на режима на изображението, превключването от текстов в графичен режим и обратно става без излизане от текстообработката, въведено е чертане на вертикални линии чрез табулатори и др.

Докато за писането на кратки текстове – писма, единични статии и др. може да се ползват и някои от по-елементарните текстообработки, то за по-сериозна работа, при писане на книги и други сложни документи МикроТЕКСТ II очевидно е извън всякаква конкуренция у нас. С него не представлява трудност да се обработват наведнъж текстове, съдържащи 500-600 000 и повече знака, особено в режим на структурирани документи, автоматично да се създава съдържание и азбучен указател, да се сортират библиографски справки (при дадените по-горе ограничения) и т.н. Продуктът притежава и безспорно най-съвършения механизъм за форматиране на текста чрез мощния и гъвкав механизъм на маските.



МикроТЕКСТ II очевидно е извън всякаква конкуренция у нас. С него не представлява трудност да се обработват наведнъж текстове, съдържащи 500-600 000 и повече знака, особено в режим на структурирани документи, автоматично да се създава съдържание и азбучен указател, да се сортират библиографски справки (при дадените по-горе ограничения) и т.н. Продуктът притежава и безспорно най-съвършения механизъм за форматиране на текста чрез мощния и гъвкав механизъм на маските.

Не е без значение и фактът, че МикроТЕКСТ II създава DOC-файлове, чийто формат е най-разпространеният и у нас, и в чужбина. Това означава, че повечето от другите текстообработващи продукти, настолни издателски системи (ИЗДАТЕЛ и РедПАК) и др. могат директно да зареждат и да работят с такива файлове, като голяма част от информацията за оформянето на текста се запазва. В комплекта на МикроТЕКСТ II са включени и помощни програми за конвертиране на DOC-файлове в текстови файлове с формàта на други текстообработки.

За повече информация и за делови контакти се обръщайте към „Микросистеми“, София 1233, ул. „Опълченска“ бл. 41А, тел. 32-90-46.

Трик за МикроТЕКСТ II

По време на работа МикроТЕКСТ II активно използва външната (дисковата) памет, като създава временни (помощни) файлове, в които съхранява направените промени и друга информация. При нормално напускане на текстообработката чрез командата Изход той записва направените промени във файла с документа и изтрива временните файлове. Често пъти обаче се случва компютърът да блокира или да се изключи без напускане на програмата, захранването да прекъсне и т.н. При всички тези случаи в справочника на МикроТЕКСТ II остава файл с разширение TMP. Попадал съм на компютри, чиито твърди дискове са задръстени с подобни файлове, което, естествено, е и атестат за квалификацията на работещите с тях.

Тъй като капацитетът на диска все не достига, временните файлове трябва редовно да се трият. Това става най-удобно, като се създаде специален справочник, например \TEMP (или с друго име) и в обкръжението на командния процесор COMMAND.COM се включи променливата TMP, на която за стойност се задава името на справочника.

Ако във файла AUTOEXEC.BAT се включи

SET TMP=C:\TEMP,

МикроТЕКСТ II ще записва временните файлове в посочения справочник.

Още по-добро решение е временните файлове да се записват в рамдиска

SET TMP=D:

Печалбата в случая е двойна:

– бързодействието на МикроТЕКСТ II допълнително се увеличава, защото рамдискът е значително по-бърз от дисковото устройство;

– след изключване на компютъра, записаните в рамдиска временни файлове се изчистват заедно със загубване на съдържанието на оперативната памет.

ЕЛЕКТРОНЕН РЪКОПИС

Авторът (аспирант в Московския полиграфически институт) прехвърля мост между текстовите редактори в компютризираните полиграфически системи. Програмата JETTYPE позволява на авторите сами да структурират текстовете си и да отбелязват изискванията си чрез управляващи символи или етикети.

ДИМО КУЦИЛЕВ

Електронният ръкопис представлява авторски оригинал, обработен на персонален компютър (ПК) чрез текстообработваща програма и записан на магнитен носител – дискета. Той може да бъде отпечатан не само върху хартия, т. е. да се получи ръкопис в неговия традиционен вид, но позволява почти мигновено издаване. Това поне внушава рекламата на настолно-издателските системи, разпространявани у нас.

Дълго време обаче надеждите на много клиенти на полиграфическата промишленост не се оправдаваха на практика. Сроковете за изпълнение на техните поръчки останаха същите, както и за ръкопис на пишеща машина. За съжаление парадната врата на издателствата и досега остава затворена почти за всичко рационално, което донесе (ПК). Истината е, че електронният ръкопис не можа да прескочи гросмайсторския срок от 4 месеца за изпълнение на наборна поръчка.

Причините са две. От една страна, текстообработващите програми имат извънредно много и различни формати на специализираните вътрешни езици за оформление. Използваната отскоро технология за конвертиране на данни представлява частично решение на проблема [1]. При тази схема значително се съкращава времето за наборните процеси, защото не се налага повторното въвеждане на авторския текст. Недостатъците ѝ обаче често пъти свеждат до нула това предимство.

При тази технология в текста се въвеждат полиграфическите команди, като операторът се ръководи от указанията на техническия редактор, отбелязани по страниците на ръкописа. А тази схема налага изданието да се контролира до окончателното коригиране на грешките, допуснати при кодирането на командите. Коректурният процес, макар и редуциран до известна степен, все още тежи върху сроковете на изданието. Ограниченото използуване на технологията на конвертиране е свързано със следните обстоятелства:

– мястото на такъв процес може да бъде само печатницата, тъй като само там познават детайлно съответната фотонаборна система;

– от специалистите, извършващи конвертирането, се изисква висока квалификация. Те трябва да познават не само фотонаборната система, но и многобройните текстообработващи програми, на които е работил евентуалният клиент. Такива специалисти в печатницата се броят на пръсти – засега като правило те са не повече от един-двама.

И двете обстоятелства подсказват, че трябва да се търси решение извън печатницата. И по-точно – да се намери отговор на следните въпроси:

- Съществува ли възможност за кодиране на текста, без да се знае как по-късно ще изглежда макетът на изданието?
- Възможно ли е да се използват всички, дори и най-обикновените текстови редактори?
- Възможно ли е така кодираните данни да се обработват изцяло машинно?

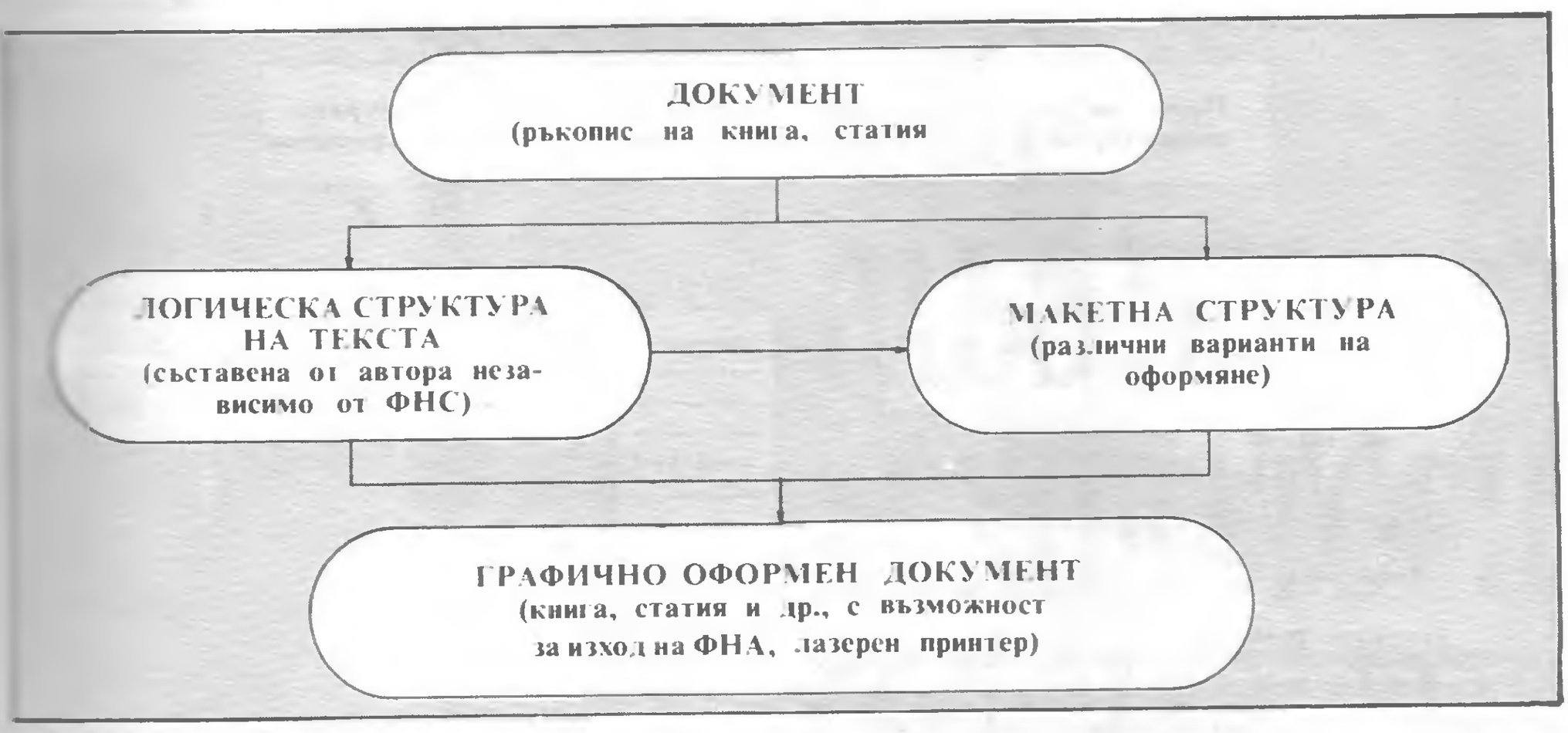
Такъв отговор дава новият програмен пакет JETTYPE, който предлагам на читателите на списанието JETTYPE представлява опит да се избегнат всички подводни камъни на полиграфическата дейност и да се осигури издаването на авторските ръкописи за максимално кратък срок. Тази статия има за цел да покаже на какви изисквания трябва да отговаря авторският оригинал, за да се автоматизира напълно обработката му независимо от разнообразието и спецификата на полиграфическите програмни и технически средства. На практика това означава:

- Преминаването на наборните процеси от тризвенна схема автор – издателство – печатница, към двузвенната автор – издателство.
- Изцяло премахване на коректурния процес.
- Съкращаване сроковете на издаването до техния възможен минимум.

В Научния център по полиграфия се разработва стандарт на базата на международния стандарт за оформление на документи SGML (Standart Generalized Markup Language) ISO 8879. Този стандарт регламентира необходимостта от разделяне на процеса на изработване на изданието на две фази: условно структуриране на текста и реализация на конкретен макет на оформление.

В първата фаза авторът сам маркира текста, отбелязвайки отделните му елементи: заглавие, подзаглавие, текст, забележка под линия и т. н. Така той изгражда логическата структура на бъдещото издание.

След това структурираният авторски оригинал може да получи различно графично оформление (1, 2, . . . , n, виж схема 1) единствено в издателството (печатницата), т. е. за графичната реализация изцяло са отговорни само графичният дизайнер на



издаването и техническият редактор.

За да се унифицира процесът по подготовката на електронния ръкопис, е необходимо създаването на единен команден език. В Научния център по полиграфия се разработва стандарт за такъв език, както и цялостна технология за създаване на авторски и издателски електронни ръкописи.

Командният език позволява предварително да се задава значението на всеки елемент от текста, като се ограждa със съответни етикети. Например заглавието на статията се огражда с етикетите <ЗАГЛАВИЕ> и </ЗАГЛАВИЕ>. Авторът може да подчертае даден блок от текст с етикети <КУРСИВ> или <ПОЛУЧЕРНО>. Краят на блока се маркира със специален маркер </>. Повече подробности за командния език може да узнаете от инструкцията за работа с програмата. Тук е важно да се отбележи, че на автора е предоставена известна свобода в избора на етикетите. Задължително е само те да бъдат оградени в специални управляващи символи – < >. А авторът заедно с ръкописа предава и текстов файл със списък на използваните етикети и кратко описание на значението им. Този подход отваря вратите за автоматизиране на полиграфическото оформление.

При подготовката на електронния ръкопис основните проблеми възникват от различния формат на запис на текстовите файлове и от ограничения брой знакове, с които разполага знакогенераторът на (ПК).

За да се избегнат конфликтите от първия род, е необходимо да спазвате следните правила.

● Записвайте информацията в неформатиран вид – в така наречения аскиформат.

● Никога не сричкоразделяйте думите в края на реда, ако вашата текстообработваща програма не ползва „меко тире“ за пренос (soft hyphen).

● Кодирайте текста в режим на безкраен ред, като означавате само края на параграфа. Някои текстови редактори не поддържат този режим. При тях е задължително краят на параграфа да се отбелязва с две последователни команди за нов ред. В някои особени случаи, ако авторът иска да подчертае края на параграф, той може да ползва етикет <ПАРАГРАФ>.

Втората група проблеми са по- многобройни. Конфликтите възникват на основата на знаковото прекодиране. Програмата може да работи с текстове, които са обработени при пълнокодова структура на кирилските знакове в аскитаблицата на компютъра. Под пълнокодова структура разбираме наличието на отделен аскикод за всички знакове на кирилската азбука – 64 на брой. На това изискване отговарят най-разпространените формати MIK, MSX, КОИ8 и др. Но трудности възникват от следния факт – знакогенераторът на компютъра поддържа 256 знака, от които 32 управляващи, докато професионалните полиграфически системи ползват над 500 знака.

Този дефицит налага допълнително кодиране за липсващите знакове. Това става чрез знака &, чието предназначение е да означава специални знакове. След този знак може да стои който и да е друг, но неговото значение се променя. Така например, ако искаме да кодираме знака & в обикновеното му значение, трябва да използваме два поредни знака &&. Точно така стои въпросът с управляващите маркери. Използвани като математически знакове в текста, те трябва да изглеждат съответно: &<, &>, &/. Във всички останали случаи авторът може сам да назначи функция на съответен знак. Например знакът „а“, кодиран със специалния символ &а, може да означава дясно ударение върху малки букви. Комбинацията &А означава дясно ударение върху малки букви. Комбинацията &А означава дясно ударение върху главни букви и т. н. По аналогичен начин се постъпва с индексите, математическите знакове, букви от гръцката азбука и т. н.

Ще разгледаме случая с тирето като по-особен.

● В знаковия генератор на компютъра има само един знак за тирето с аскикод 45. Той се използва за всичко – пренос на думи, за пряка реч, за разделяне на сложни думи и др. Затова по някакъв начин авторът трябва

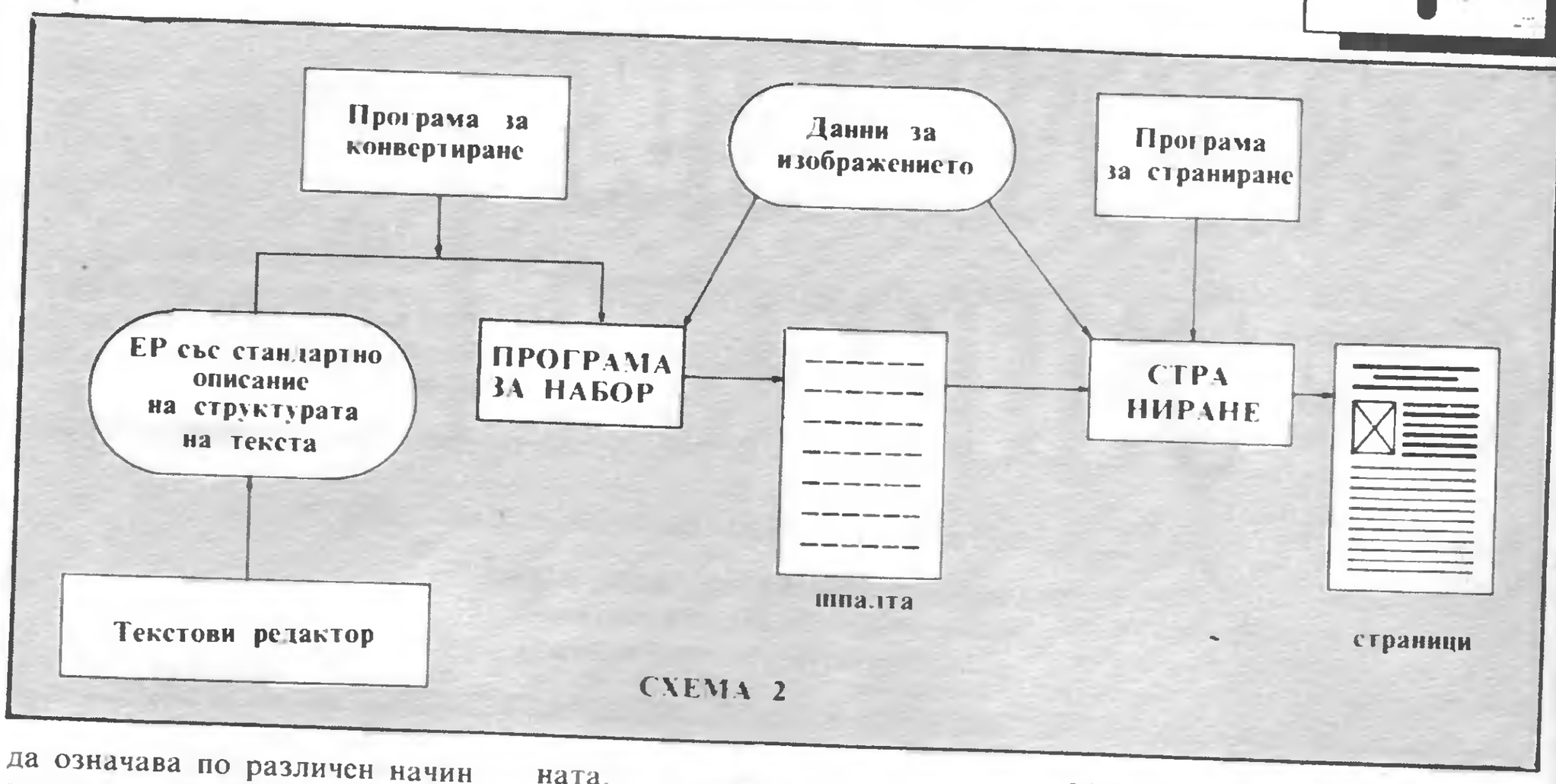


СХЕМА 2

да означава по различен начин тирето и късото тире (дефис). В моята програма е възприет следният подход – за малкото тире се използва аскикод 45, а тирето се кодира с два знака с аскикод 45. И понеже математическият знак минус изглежда по същия начин, за да избегнете всякакво двусмислие, кодирайте го в текста с &–.

● Ако се налага да използвате един и същ знак в различен функционален смисъл, например цифра в ролята на индекс, използвайте един от двата възможни начина: означете я със специалния символ & или сложете етикет.

● Не използвайте никога директно математическите знакове, дори когато съществуват в аскитаблицата. Означавайте ги като специални.

● Избягвайте ръчните или автоматични възможности за сричкоразделяне на текстообработващия продукт, с който работите.

● Структурирайте колкото е възможно по-пълно отделните елементи на вашия текст. Означавайте тяхното начало и край.

И естествено не забравяйте да предадете заедно с ръкописа списък на всички използвани от вас етикети и специални знакове. За целта можете да създадете текстови файл, в който да опишете значението имнеформално.

Организацията на работа с програмата JETTYPE е следната.

Основният модул JETTYPE представлява програма за прекодиране (вж. схема 2) на текстови файлове с аскиформат, специализиран формат на съответната текстообработваща система или изходно устройство.

Най-напред в програмата се зарежда драйверът за съответното устройство (понятието устройство се интерпретира в най-широк смисъл). След това се посочват имената на входния файл в аскиформат и на изходния файл, който се стремим да получим. При неправилен формат на данните програмата посочва грешката, но в тази версия тя може да се коригира само след излизане от програмната среда.

Драйверните модули имат текстов формат и представляват богатството от възможностите на JETTYPE. Техните имена притежават разширение .EMU. Всеки специалист, познаващ добре системните възможности на професионалната текстообработваща система, независимо каква е тя, може да създаде драйвер за нея. Така програмата ще обхване всички печатници работещи с фотонабор, естествено при евентуален успех.

В този вид JETTYPE е подготвен за издателствата и печатниците, работещи с текстообработващата система POLTYPE. Надяваме се, че специалистите ще се заинтересуват и скоро ще бъдат създадени драйвери за текстообработващите системи MILES и LINOTYPE. Практическото използване на програмата JETTYPE показва безспорните предимства пред обикновеното конвертиране на данни. При по-нататъшното отработване на схемата и постепенно натрупване на производствен опит ще се използват готови драйвери – шаблони, за няколко най-често прилагани макети на издания. Така ще бъде постигната пълна автоматизация на процесите, наистина без никакъв ръчен труд.

В момента се работи по създаване на програма, която ще позволява прекодирането на авторския текст да се извършва на компютри с непълнокодова структура на аскитаблицата. Това означава, че електронни ръкописи ще се изготвят и на широко разпространените у нас осембитови компютри.

ЛИТЕРАТУРА

1. Хьорл Х., Перспективи и развитие на обработка на текстова информация. Полиграфия, (1988), No 3.

2. Панчева Р., Връзката електронен фотонабор-компютри в издателската дейност. Полиграфия, (1988), No 3.

Бележка на редакцията. Програмата Jettype се намира в редакцията на разположение на читателите на списанието и всеки може да си я запише. Необходима е една дискета..

отбележа, преди да опиша алгоритъма за „декомпресия“ на псевдокод, е, че съставената при компресията таблица не е нужно да се предава като спомагателна информация заедно с компресирания файл. Тази таблица се възстановява от „декомпресиращия“ алгоритъм в процеса на възстановяване на оригиналния файл. Това е възможно, тъй като компресиращият алгоритъм винаги изпраща в изходния файл кода на напълно разпознатия „STRING“, преди да го използва при образуването на нов „STRING“. И така алгоритъмът за възстановяване е следният:

```
програма LZW_възстановяване
OLD_CODE = getchar( от входен файл );
запиши OLD_CODE в изходния файл;
WHILE ( не е достигнат края на файла  )
        {   /* тяло на алгоритъма */
        NEW_CODE = getchar( от входен файл );
        STRING = преобразуване на кода NEW_CODE в символен низ;
        запиши STRING в изходния файл;
        CHAR = първият символ от STRING;
        прибави OLD_CODE + CHAR към преобразуваната таблица;
        OLD_CODE = NEW_CODE;
        }
```

Тук функцията getchar( ) само илюстрира четенето от входния файл. Прочетените стойности не се вместват в един байт (както например кодовете от 256 нагоре в примера за компресирането). Също както компресирането възстановяването строи таблица и прибавя нов символен низ при прочитането на нов код. Може да възникне законен въпрос: защо „STRING“ таблицата започва от 256? Тези, които си спомнят първата част, може би веднага ще си отговорят, защото първите 255 стойности са заети от едносимволните „аскинизове“. Сега вече сме готови да възстановим символната редица от получения след компресията вид:

Кодове: .A B C 256 B 260 261 257 D 260 E

От посочените кодове възстановяващият алгоритъм ще даде следния резултат:

четене на NEW_CODE от входен файл	стойност на OLD_CODE	STRING в изходен файл	CHAR	кодова таблица
	.	.		
A	.	A	A	256 = .A
B	A	B	B	257 = AB
C	B	C	C	258 = BC
256	C	.A	.	259 = C.
B	256	B	B	260 = .AB
260	B	.AB	.	261 = B.
261	260	B.	B	262 = .ABB
257	261	AB	A	263 = B.A
D	257	D	D	264 = ABD
260	D	.AB	.	265 = D.
E	260	E	E	266 = .ABE

Възстановеният изходен файл е (както може да се види от STRING):

.ABC.AB.ABB.ABD.ABE<EOF – край на файла>

„Странно“, нали? Оригиналният и възстановеният файл са еднакви. Таблиците също! По-малко не може и да се очаква от един на пръв поглед объркан, но в крайна сметка изящен алгоритъм. Както и преди, няма да е зле да се извърши „ръчно“ проследяване. За улеснение на такова проследяване ще отбележа, че:

- Всяка операция четене придвижва показалеца на файла върху следващия обект (при компресия символ, при възстановяване код с определена предварително дължина).

- Първият прочетен код се появява на изхода, без да се присвоява на STRING, но в таблицата това първо извеждане е дадено под STRING, тъй като по принцип изходният символен низ се получава в тази колонка.

- На един и същ ред от таблицата на примера стойността на OLD_CODE е получената от предишна итерация (или от началото), докато стойността на NEW_CODE е получена от текущата итерация (четене).

Дотук всичко изглежда идеално за тандема компресия-възстановяване. С изключение на един-единствен случай. Да предположим, че някъде в таблицата на компресията вече е бил дефиниран символен низ (и съответният код) от вида (STRING, CHAR). Ако някъде по-нататък има комбинация от вида STRING, CHAR, STRING, CHAR, STRING, компресиращият алгоритъм ще подаде на изхода си код, който възстановяващият алгоритъм не би могъл да дефинира в собствената си таблица, тъй като още няма да е стигнал до това място от нея. Например някъде при компресията е дефиниран символен низ ABCD с код 274 (числото не е някакво специално, а е резултат от „ръчното“ проиграване на алгоритъма върху малък символен низ /около 90 байта/). След това се среща поредицата ABCDABCDABC. Първата част ABCD води до подаването на код 274 на изхода.

След това в процеса на компресията се образува низ ABCDA, на който се приписва поредният кодов номер /299/. Този код се подава на изхода веднага след 274, тъй като в процеса на компресията следва образуването на ABCDAB. При възстановяването от код 274 на изхода се записва символният низ ABCD, както и би трябвало. Като се следва алгоритъмът на възстановяване, от новия STRING се взема първият символ и се прибавя в края на символния низ, дефиниран от OLD_CODE. По този начин се дефинира нов елемент от таблицата. След това се прочита код 299, но за този код още не е създаден елемент от таблицата. Този случай за щастие е единственият, при който може да възникне неопределеност. Тази неопределеност се премахва със следната модификация на основния алгоритъм:

Премахва се инструкцията от псевдокода

```
STRING = преобразуване на кода NEW_CODE в символен низ:
         и на нейно място се поставя,
IF NEW_CODE не е в таблицата на преобразуването THEN
        STRING = преобразуване на кода OLD_CODE в символен низ;
        STRING = STRING + CHAR;
ELSE
        STRING = преобразуване на кода NEW_CODE в символен низ;
ENDIF
```

С това завършва неформалното и необременено с детайлите на конкретна реализация (програма на Си, Асемблер, Паскал или друг някой език) описание на компресиращите Lempel/Ziv/Welch (LZW) алгоритми.

По-долу ще бъде описан методът на вероятностното кодиране по Хъфман. Този метод се основава на теоремата на Шенон за кодирано предаване на информация по канал без шумове. Тази теорема гласи, че ако в канала за връзка няма шум, то може да се намери такъв метод за кодиране, при който средният брой на двоичните цифри за предаване на единица информация може да се доближи произволно близко до ентропията на информационния източник, но не и да бъде по-малък от нея. На практика това означава, че на често срещащи се информационни единици (например байтове с определена стойност) може да се приписват по-кратки кодови комбинации, а на по-рядко срещащите се — по-дълги. Така че, за да компресираме даден файл по този метод, първо трябва да го прочетем изцяло и да преброим колко пъти се среща всяка кодова комбинация (в нашия случай байтове). След това от образуваната таблица се избират символите, които се срещат най-рядко.

Тези символи се свързват с помощта на структура, наподобяваща двоично дърво. Мястото на връзката се нарича възел и при програмната реализация към него сочи указател. Целият метод ще бъде пояснен въз основата на пример. Нека имаме 100-байтов файл, в който има общо 6 различни символа. След като сме намерили символите, преброяваме колко често се среща всеки от тях. Така стигаме до следната таблица:

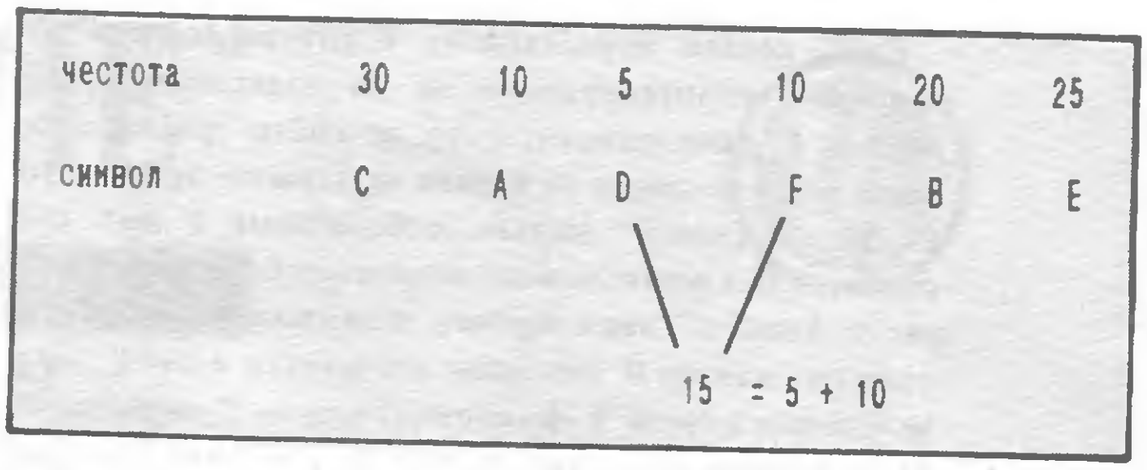
КОЛКО ПЪТИ,	СИМВОЛ
5 -	D
10 -	A
10 -	F
20 -	B
25 -	E
30 -	C

За да е по-нагледно образуването на дървото, ще представим таблицата в следния вид:

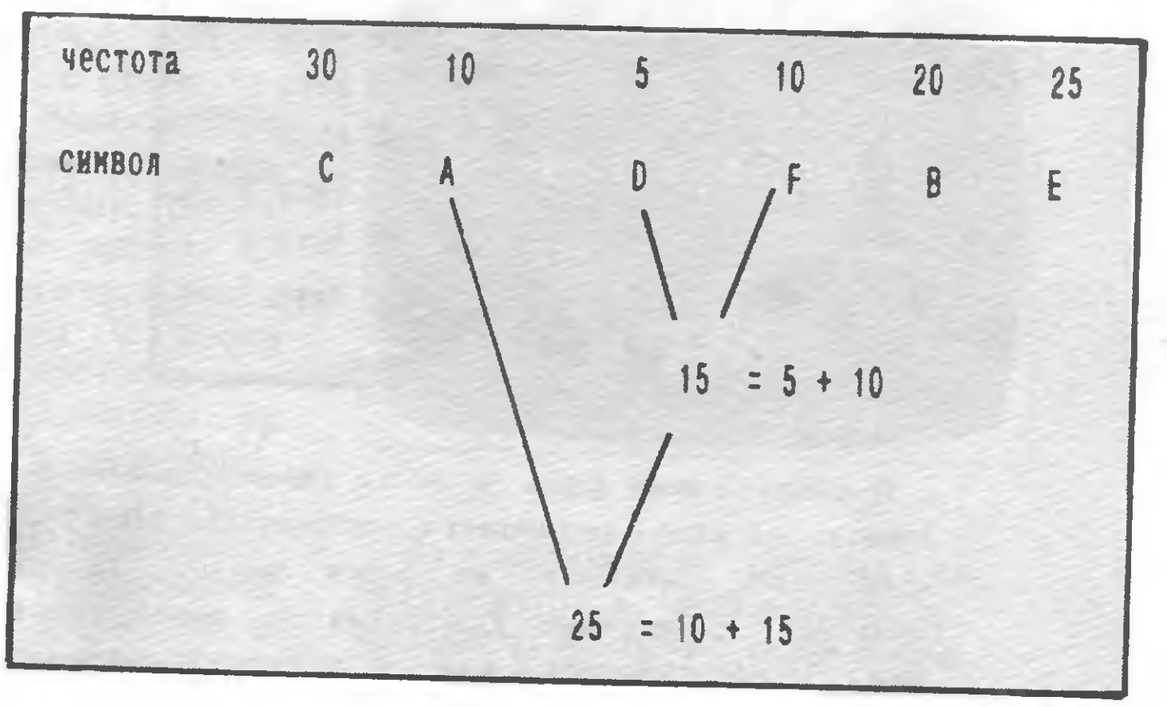
честота	20	10	5	10	30	25
символ	B	A	D	F	C	E

След това изследваме таблицата, за да намерим символите с най-малка честота (т. е. тези, които се срещат най-рядко). Ясно е, че единият от тях ще е символът с честота 5, но кой от двата с честота 10 да изберем?

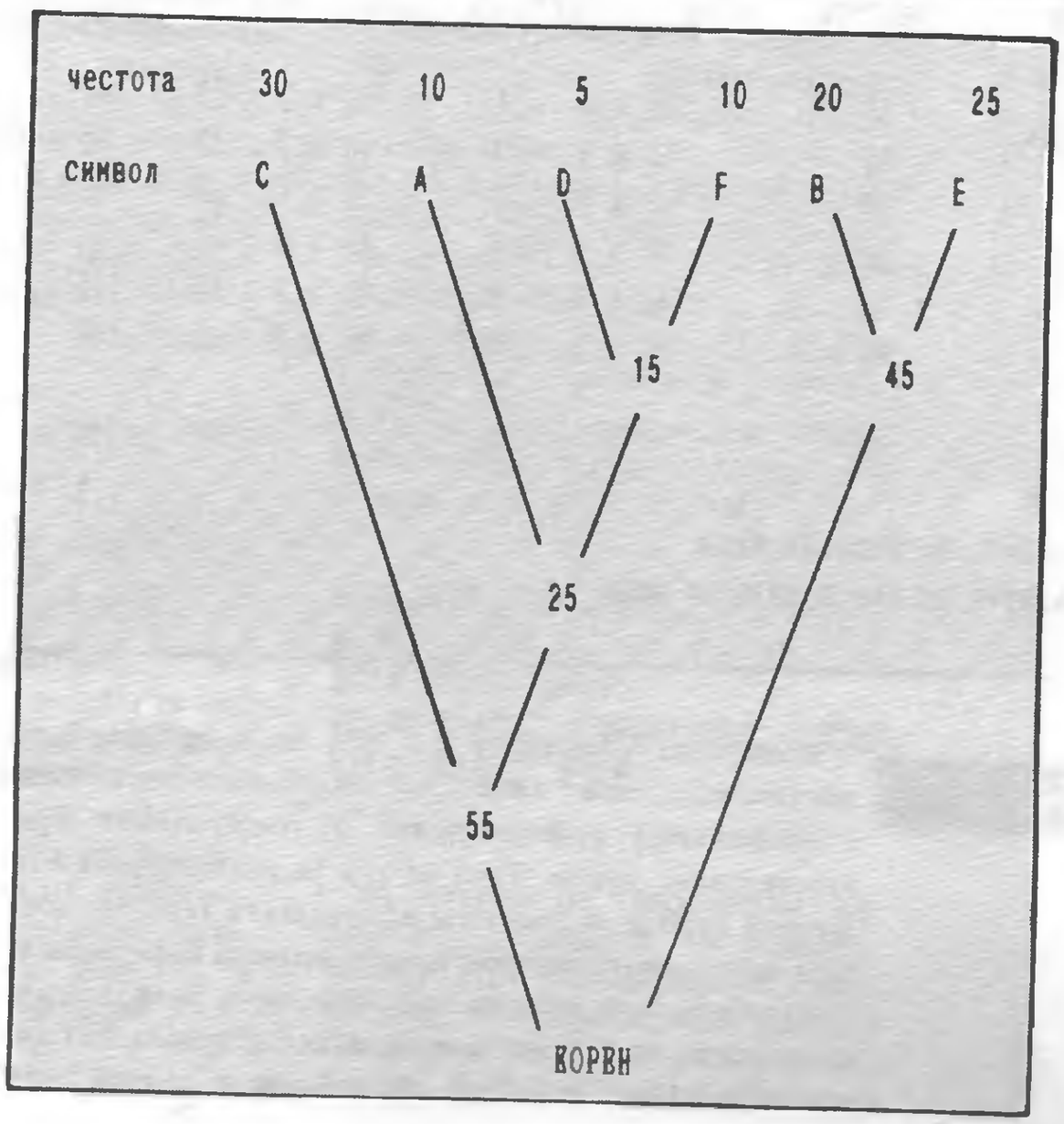
Теорията учи, че това няма особено значение, и ние произволно ще изберем. Сега свързваме F и D така, че да се образува възел на дървото.



Числото във възела е сума от честотите на F и D. Продължаваме да изследваме таблицата, като вече изключваме от търсенето F и D. Следващите най-малки числа са 10 за A и числото на образувания преди малко възел 15. Отново конструираме част от дървото:



След това виждаме, че следващите кандидати са B и E. Продължавайки по този начин, построяваме цялото дърво, т. е. стигаме до корена. Листата на дървото са самите символи.



Сега, когато вече дървото е готово, можем да започнем компресирането. За да кодираме първия символ С, проследяваме пътя, по който трябва да се мине, за да се стигне от корена на дървото до листото С. Всеки „завой" наляво отбелязваме с бит със стойност 0, а всеки „завой" надясно отбелязваме с бит със стойност 1. Така например за символа С записваме: един път наляво 0, още един път наляво и пак 0, така че в нашия случай Хъфмановият код на С ще бъде 00. За А вървим един път наляво 0, един път надясно 1 и още един път наляво 0. Извършваме описаните действия за всички символи и получаваме следната таблица:

```
C = 00        ( 2 бита )
A = 010       ( 3 бита )
D = 0110      ( 4 бита )
F = 0111      ( 4 бита )
B = 10        ( 2 бита )
E = 11        ( 2 бита )
```

В оригиналния файл за всеки символ имаме по 8 бита. Тъй като с кодирането, което извършихме тук, намалихме дължината на всеки символ, го на практика извършихме компресия на оригиналния файл. За ефективността може да се съди от следната таблица:

символ	честота	битове в оригинала	компресия	"печалба"
C	30	30 x 8 = 240	30 x 2 = 60	180
A	10	10 x 8 = 80	10 x 3 = 30	50
D	5	5 x 8 = 40	5 x 4 = 20	20
F	10	10 x 8 = 80	10 x 4 = 40	40
B	20	20 x 8 = 160	20 x 2 = 40	120
E	25	25 x 8 = 200	25 x 2 = 50	150
Общо	100	800	240	560

дължина на входния файл
дължина на компресирания файл

Идеалната компресия от 70% не може да бъде достигната, тъй като за възстановяването ще е необходима информацията от построеното при компресията дърво. Така че тук за разлика от LZW метода трябва да запишем кодиращата таблица, или по-точно дървото заедно с компресираната информация. Декодирането е тривиално. Тъй като разполагаме с дървото, при прочитане на всеки следващ бит от компресирания файл ще знаем какъв „завой" по дървото да направим. В крайна сметка този процес ще ни отведе до листо, в което се намира необходимият символ. Като намерим символ, т. е. го възстановим, преди да прочетем следващия бит, отново се връщаме в началото на дървото. След това всичко се повтаря. Ако се ограничим с 256 възела за дървото и използваме за всеки възел по 4 байта за указатели, пълната таблица ще заеме 1 Кбайт. В нашия пример имаме 5 възела и 6 листа, или общо 11 места за указатели. Това прави 44 байта, към които ще прибавим и 6 байта, в които са записани съответните честоти. Тези 50 байта ще прибавим към информационните 30 (240 бита / 8 бита на байт). По този начин получаваме фактор на компресия 20%. Наименованието вероятностно кодиране може да се струва на някого не съвсем точно. Не трябва да забравяме обаче, че тук все пак в основата лежи определено статистическо измерване на честотата на появяването на даден символ. Теоретичният максимум, който може да се постигне при Хъфмановото кодиране на символна (байтова) основа, е 33% компресия. Трябва специално да се отбележи, че описаните в двете статии методи за кодиране са много чувствителни към грешки. Например промяната на един бит в Хъфмановия код означава най-малкото промяна в пътя по дървото при декодирането, а оттам и до пълно объркване. Това естествено е нежелателно и в реалните системи никога не се разчита на липсата на шум в канала за връзка, респективно на бездефектен носител на информацията (флопи- или харддиск). За съжаление единственото надеждно кодиране е кодирането с излишък, т. е. обратното на компресията. Все пак изход има. Кодирането с излишък не се прилага към всяка информационна единица, а към цял блок от такива единици (например байтове). Съществуващото огромно множество на кодове с излишък, които сами откриват грешки и дори ги премахват, позволява да се постигне разумен компромис между дължината на кодирания информационен блок и необходимия за надеждно съхраняване и пренасяне информационен „излишък". Най-популярни в архиваторите са CRC кодовете (от Ciclic Redundan check Code – Цикличен код с излишък). Теорията на образуването на CRC кодовете е достатъчно сложна, но може би в някой следващ брой на списанието ще успея да го представя по начин, удобен за лесно и бързо „разгадаване". Освен това има и други алгоритми за компресиране като например аритметичното кодиране, които може би също ще намерят място на страниците на списанието.

КВ–28

ОБЯВА

ТЪРСЯ притежатели на компютър Комодор—64 за размяна на програмни продукти.

Мартин Иванов,
8000 Бургас,
ул. Цар Калоян 15,
ет. 3, тел. 056—4-55-69.

„ОТПЕЧАТВАНЕ“ НА ЕКРАНА ВЪВ ФАЙЛ

СТЕФАН КОЖУХАРОВ

Отпечатването на екрана върху принтер е една отлична възможност да се получи копие на някои таблици или резултати от изпълнението на програма. Често обаче се налага редактиране на тази информация, което за съжаление е невъзможно при директното отпечатване върху хартия. Например искаме да напишем ръководство за работата с програма, която използва прозорци. В този случай преснимането на екрана или на отпечатъка не е най-сполучливото решение. По-добре е да се напише програма, която остава резидентна в паметта и позволява записването на екрана във файл. Това дава възможност за редактиране на получената информация, а освен това текстът може лесно да се вмъкне на необходимото място.

Предлаганата тук програма има и някои допълнителни възможности: начертаване на рамка по образец, зададен от потребителя; удебеляване на инверсните знакове и други.

Стартирането на програмата се извършва чрез WSC.

След стартирането на програмата натискането на клавишите Shift-PrtSc предизвиква записването на екрана във файл с име SCREEN. TXT. Новите екрани се добавят към съществуващите, като на края на всеки екран се добавя знак за нова страница. В тази версия не е предвидена възможност за запис на графични екрани. По желание програмата може да се разшири така, че това ограничение да отпадне. При работа с тази програма е предвидена възможност инверсните знакове да се отпечатват с двоен печат, което дава възможност да се направи разлика между инверсни и обикновени букви. Много често при работа се използват графичните знакове лява, дясна, долна и горна стрелка. При директното отпечатване тези знакове се възприемат като управляващи и може да се стигне до промяна на режима на печатащото устройство. Програмата автоматично ги замества с последователности от знакове, които дават приблизителен аналог на стрелките.

Една полезна възможност е екранът да се отпечата в рамка. Тъй като програмата остава резидентна, за предаване на необходимите за начертаване на рамката знакове се използва обкръжението. В него се създава една променлива, наречена BOX. Това се извършва чрез следните команди:

SET BOX=*последователност от знакове*

или

SET BOX=*„последователност от знакове“*

Последователността от знакове започва от горния ляв ъгъл и продължава по посока на часовниковата стрелка. Втората команда се използва, когато се задава последователност от управляващи символи Например, за да получим рамката, показана на фиг. 1, е необходимо да се напише:

SET BOX=„+−+−+−“



фиг. 1

За правилната работа на програмата е необходимо задаването на точно осем знака.

Напрмер:

SET BOX=┌┐│┘─└│

Програмата е написана на Асемблер за микропроцесор 8086, като са спазени изискванията за построяване на .COM файлове. Това дава възможност за преобразуване на програмата от .EXE в .COM формат, което ускорява зареждането ѝ. Получаването на работеща програма се извършва чрез следната последователност от команди към операционната система:

```
MASM /M1 WSC;
LINK WSC;
EXE2BIN WSC;
REN WSC. BIN WSC. COM
DEL WSC. EXE
```

И накрая да проследим накратко работата на програмата. При стартиране найнапред тя проверява дали в паметта вече има резидентно копие. Това се извършва чрез

прехващане на INT2F. Ако гаденият от INT2F отговор съответства на избраната последователност от знакове, не се извършва инсталиране на второ копие. В противен случай INT5 (Отпечатване на екрана) се заменя с нова програма за обработка на прекъсването и се прехващат прекъсвания INT21 и INT2F. С това първият етап от работата на програмата приключва и тя остава резидентна в паметта.

С натискане на Shift-PrtSc започва вторият етап от нейната работа. Най-напред трябва да се направи проверка, дали програмата е приела предишна заявка за печат. Тъй като заявката за печат може да дойде по всяко време, необходима е проверка, дали ДОС има свободни ресурси за работа с файлове. Ако ДОС не е способна, само се вдига флагът за заетост и с това приключва и вторият етап от работата на програмата. Необходимостта да се изчака освобождаването на ДОС наложи изискването да се натисне клавиш, когато ДОС чака въвеждане на низ (а също и когато ДОС чака въвеждане на команден ред).

Ако ДОС е свободна, се преминава към третия етап от работата на програмата — същинското записване на екрана във файл. През този етап се преглежда обкръжението за променливата BOX. Ако тя съществува, се изчертава рамка, а ако не — премахват се десните интервали.

Бележка на редакцията:

Поради дългия текст на програмата нямаме възможност да я публикуваме. Но всички желаещи могат да дойдат в редакцията, за да си я презапишат.

КВ–30

РЕЦЕНЗИЯ

ПРОГРАМИРАНЕ НА АСЕМБЛЕР

Предлаганата книга е за начинаещи програмисти, но все пак се изисква някаква предварителна подготовка: би било добре читателят да има опит в програмирането на езици от високо равнище например като Бейсик или Паскал. За щастие това изискване не е съвсем строго – изложението е популярно и позволява обучение и без такъв опит.

Книгата започва с устройството на персоналния компютър IBM PC и описание на езика асемблер за процесорите 8086/88. Авторът разглежда подробно двоичната аритметика, машинния език и асемблера, действието на компютъра, основни принципи на програмирането.

Разгледан е подробно микропроцесорът 8088: програмен модел, регистри, адресация.

В глава 4 е дадена системата от инструкции за микропроцесора 8088. В главата за дисковата операционна система (ДОС) са разгледани файловата система, командният интерпретатор, функциите на ДОС, командни и изпълними файлове, съставяне програми на асемблер, редовият редактор на ДОС, асемблер и макроасемблер, свързваща програма, програма за коригиране и преобразуване на изпълними файлове.

По-нататък подробно са разгледани възможностите на програмата макроасемблер, аритметичния процесор 8087, устройството на персоналния компютър IBM PC, служебната програма за базов вход-изход (BIOS), подпрограми и разширения на служебните програми на асемблер, системата от инструкции за микропроцесора 8087.

ВИЛХЕЛМ ИВАНОВ

1) Брадли, Д. Програмиране на АСЕМБЛЕР за персонален компютър IBM/PC. Прев. от англ. София, Техника. 1989, 310 с.

Обяви

ТЪРСЯ притежатели на Правец—8Д за размяна на литература, игри и програми.

Светослав Кръстев,
2600 гр. Станке Димитров,
ул. Чайка 6,
тел. 0701—2-67-43 след 18 часа.

● ● ●

ТЪРСЯ помощ за записване на програми за Правец—8Д. Ще изпратя веднага касети на този, който би ми записал игри, програмата „Монитор" и други.

Георги Георгиев,
3700 Видин,
тел. 094—3-02-18.

● ● ●

Предлагам собствени игри за АТАРИ—СТ.

Теодор Кръстев
София
тел. 58-96-39

ТЪРСЯ притежатели на компютър SHARP MZ–700 или съвместим за размяна на програмни продукти.

Бяндов,
4000 Пловдив,
тел. 032—23-38-42.

● ● ●

ПРОДАВАМ компютър ЕПЪЛ IIe с дисплей и дискетно устройство.

Вл. Хлебаров,
9000 Варна,
кв. Чайка,
бл. 26, вх. А, ап. 27,
тел. 052—88-84-12.

● ● ●

Търся литература и програмни продукти за ПРАВЕЦ—8Д.

Слави Стойков
Елин Пелин
ул. Мине Иванов 25
бл. ДК, вх. А, ап. 10
тел. дом. 99-71-51/ 36-84

НАПРЯКО

ПРЕЗ БАЙТОВЕТЕ

Н. с. инж. ВЕСЕЛИН БОНЧЕВ

В своята практика често пъти съм срещал различни хитри програмистки похвати. Тези „хватки“ не са описани в ръководствата, но в тях няма никаква „магия“. За да ги измисли, на човек са му нужни само отлично познаване на споменатите ръководства и, разбира се, малко умствена гимнастика. Веднъж измислени обаче и показани на неосветения, тези трикове предизвикват спонтанна реакция — най-често пляскане по челото и възклицание от рода на „Как не съм се сетил досега!“ или по-примитивното „И-ха-а!“

Щатата настрана, повечето от триковете са доста полезни във всекидневната практика, а разбирането на начина им на действие води до по-задълбочено разбиране и на тънкостите на конкретната операционна система или транслатор.

Ето няколко хватки, групирани според операционната среда, за която са предназначени. За съжаление в повечето случаи авторът им не може да бъде установен — те се предават от уста на уста и от компютър на компютър като програмистки фолклор.

Бейсик за Правец-82

Привържениците на структурното програмиране (доколкото те изобщо пишат на Бейсик) могат да внесат свежа струя в листингите си. Програмите им ще придобият значително „по-паскалски“ вид, ако бъдат оформени, както е показано на листинг 1. Този похват значително подобрява четливостта на програмите, но за съжаление влошава бързодействието и увеличава заеманата от тях памет.

```
10  FOR I = 1 TO 3
20 : FOR J = 1 TO 3
30 :: FOR K = 1 TO 3
40 ::: PRINT I,J,K
50 :: NEXT
60 : NEXT
70  NEXT
```

Листинг 1

ПроДОС

За да определите началния адрес и дължината на даден файл в ДОС 3.3, е необходимо да го заредите в паметта и да проверите съдържанието на клетки $AA72, $AA73 (43634, 43635) и $AA60, $AA61 (43616, 43617). Ами ако няма достатъчно свободна памет? Или ако ви трябва да знаете и типа на файла? И в двата случая няма да минете без сложни програми, които извличат информацията директно от каталога на дискетата, като го четат по сектори.

В ПроДОС всичко това може да стане много по-просто. Изпълнете командата

VERIFY име_на_файл, където име_на_файл е името на необходимия ви файл. След това получавате:

На адреси	
48824	Типа на файла
48825, 48826	Адреса, от който би се заредил в паметта
48828, 48829	Дължината му в байтове

Двубайтовите числа (адресът и дължината) са разположени по познатата схема — младшият байт е на по-малкия адрес, а старшият — на по-големия.

Паскал за Правец-82

Ако в една знакова променлива, да речем с име CH (от тип char) се съдържа редовна буква от латиницата ('a'..'z'), кой е най-бързият начин да я превърнем в главна ('A'..'Z')? Във всеки случай не операторът .

```
CH:=CH-('a'-'A');
```

Отговорът е .

```
CH:=chr (ord (odd (ord (CH)) and odd (223)));
```

но той вероятно се нуждае от пояснения. Работата е там, че функциите chr, ord и odd не генерират код, а се изчисляват от компилатора. По-точно, те служат за превръщане на типовете:

Функция	Превръщане
chr	integer->char
ord	char->integer, boolean->integer
odd	integer->boolean

Така че на практика нашият оператор върши следното:

– превръща съдържанието на променливата CH в integer

– превръща резултата в boolean

– превръща числото 223 ($DF) в boolean

– извършва операцията логическо И между двата резултата (единственият оператор, който генерира код)

– превръща резултата от boolean в integer

– превръща резултата от integer в char

– присвоява го на променливата CH (тук също се генерира код)

□ □ □

На програмистите на Паскал, които искат да съкратят размера на програмите си и да ги направят по-бързи, може да се дадат още няколко съвета:

– не използвайте оператор case с големи интервали между съседните стойности на селекторите. Например

```
case NUM of
        1:action1;
     1000:action2
end;
```

генерира твърде много код. Заменяйте такива оператори с еквивалентните им последователности от if...then...else if...

– избягвайте използването на междинни променливи (т. е. такива, които не са локални за дадена процедура, но не са и глобални за цялата програма).

– след като настроите програмата си, компилирайте я отново с опциите към компилатора (*$R–, I–*). Те изключват съответно проверките за излизане извън декларираните граници на масивите и за входно-изходни грешки. Това може да съкрати до 30% генерирания код!

□ □ □

За да се запише някаква стойност на определен адрес от паметта, обикновено се прави следното (листинг 2). Но това може да се направи по-кратко, по-лесно, по-бързо и по-елегантно (листинг 3).

□ □ □

При деклариране на тип диапазон вместо знака „..“ може да се използва по-късото „:“.

CP/M

Когато изпълнява външни команди (т. е. програми, записани във файлове с разширение .COM), операционната система CP/M винаги ги зарежда от един и същ адрес – 100H. Това подсказва мисълта, че ако след завършване на изпълнението на програмата предадем управлението отново на този адрес, ще предизвикаме рестартирането ѝ.

При това без да е необходимо да изчакваме мудното ѝ зареждане от дискетата, която най-вероятно междувременно вече сме сменили.

Как да постигнем това? Много просто – създайте файл с разширение .COM и нулева дължина.

Това може да стане чрез командата

```
SAVE 0 RUN.COM
```

Файлът RUN.COM не заема място на дискетата, а само в директорията ѝ. Когато го изпълнява, CP/M го „зарежда“ от адрес 100H и му предава управлението. Но на този адрес все още се намират инструкциите на последната стартирана външна команда.

За да проверите действието на новата „команда“, изпълнете някаква програма, например STAT.

След това въведете RUN. Резултатът ще бъде същият. Новата команда може да приема дори параметри: опитайте с

```
RUN *.*
```

ДОС – 16

В операционната система ДОС – 16 няма команда, аналогична на командата VERIFY в ДОС 3.3 за Правец-82. Поточно, команда VERIFY има, но тя върши нещо съвсем друго.

Ако все пак искате да разберете дали файловете от една дискета се четат без грешки, можете да използвате командата

```
copy a:*.* nul
```

(Предполага се, че дискетата е в устройство A:. Ако на нея има и поддиректории, горната команда трябва да се изпълни за всяка една от тях.)

□ □ □

Много хора се чудят за какво може да се използва името „.“, което се намира във всяка поддиректория. Ето един пример за употребата му.

Представете си, че имате много натоварена променлива PATH, а програмата, която искате да

стартирате, със сигурност се намира в текущата директория. Да предположим, че искате да изпълните BAT-файл и името му е PROGRAM.BAT. За да го стартирате, достатъчно е да въведете program от клавиатурата. Но да предположим, че сбъркате и напишете

progam. Тогава ще се извърши следното:

– текущата директория ще се претърси за файл с име

PROGAM.COM;

– текущата директория ще се претърси за файл с име

PROGAM.EXE;

– текущата директория ще се претърси за файл с име

PROGAM.BAT;

– тъй като, по всяка вероятност, нито един от тях няма да бъде намерен, горните точки ще се изпълнят за всяка от описаните в променливата PATH директории;

– едва след всичко това ще се изведе познатото съобщение „Bad command or file name". А ако на всичкото отгоре и твърдият ви диск не е от най-бързите...

Достатъчно е обаче да укажете, че файлът трябва да се търси само в текущата директория (т. е. да не се използва променливата PATH), и описаният процес ще протече значително по-бързо. А това става така:

program

□ □ □

Известно е, че командите

dir dir1 и

dir dir1*.* са еквивалентни, ако разбира се, \DIR1 е поддиректория. Това обаче важи не само за dir, но и за командите del и erase. Например

del dir1 изтрива всички файлове в поддиректорията \DIR1.

Между другото, това означава, че вместо известното

del *.* можете да използвате със същия успех по-краткото

del

□ □ □

Налагало ли ви се е някога да изтриете всички файлове от една директория чрез команда на .BAT-файл? Не можете просто да напишете

del *.* тъй като на екрана ще се появи досадното (и доста излишно според мен) съобщение

Are you sure (Y/N)?

При което обработката на .BAT-файла спира, докато потребителят не въведе „y" или „n", последвано от клавиша RETURN. Ами ако .BAT-файлът е инсталираща програма, предназначена за потребители, които не само че хабер си нямат за това, което става в момента, но дори не знаят английски?

Изходът е да се използва следният програмен канал:

echo y | del. > nul

Командата echo осигурява необходимото „y" на командата del, а съобщението „Are you sure (Y/N)?" се пренасочва към специалния файл nul и не се появява на екрана.

□ □ □

Във версиите на PC-DOS преди 2.0 текстовите файлове задължително трябваше да завършват със знака Z (Ctrl-Z, ASCII-код 26 или 1Ah). В следващите версии всички уважаващи себе си програми не се интересуват от последния байт, а използват служебната информация за дължината на файла, записана в директорията. За съжаление командата type работи „по старому".

Да предположим, че файлът FILE съдържа текста „Line 1", следван от знака ^Z. Ако към него добавим текст, например чрез

echo Line 2>>file

а след това изпълним

type file

на екрана ще се появи само първият ред от файла. Причината за това е, че командата type спира при срещане на знака ^Z.

```
CONST
   C030 = -16336;        { $C030 }

TYPE
   Byte = 0:255;
   Magic = RECORD
              CASE Boolean OF
                 True : (Addr : Integer);
                 False : (Ptr : ^Byte)
           END;

VAR
   Mix : Magic;

BEGIN
   Mix.Addr := C030;
   Mix.Ptr^ := 0
END.
```

Листинг 2

```
CONST
   C030 = -16336;        { $C030 }
TYPE
   Byte = 0:255;
   Memory = PACKED ARRAY [0:0] OF Byte;
VAR
   Ptr : ^Memory;
BEGIN
   Ptr := NIL;
   {$R-}
   Ptr^[C030] := 0
   {$R+}
END.
```

Листинг 3

Проблемът може да се реши, като този знак се премахне от файла. Това става чрез командата

copy file+nul file /b >nul

Отделните части на командата изискват може би по-подробно обяснение. Основната идея е файлът да се прочете в ASCII вид

(опция /a на командата copy) – тогава ^Z се пренебрегва – и да се запише в двоичен вид (опция /b) – тогава на края му не се добавя ^Z. Опцията /a е изпусната, тъй като за файлове, свързани със знака „+“ тя се подразбира. Естествено би могло да се запише просто

```
copy file /a file /b
```

но командата copy е достатъчно „хитра“ и не допуска даден файл да бъде копиран в себе си.

(Как точно става това, е тема на отделен разговор.) Затова се налага тя да бъде „излъгана“ чрез използването на специалния файл nul.

Ще добавя само, че стандартният изход се пренасочва в nul, за да се избегне появата на съобщенията

```
FILE
NUL
1 File(s) copied
```

Изобщо тук се използва забележителното свойство на файла nul – при четене от него възниква ситуацията „край на файла“, а при запис – не се извършва никакво действие, освен че се връща код за успешно завършване на операцията.

Б. Р. Ако вие, уважаеми читатели, сте срещали други програмни трикове, които ви струва да бъдат широко използвани, споделете ги с останалите читатели.
Адресът на редакцията е известен:

София 1000
бул. Толбухин 51 А
Компютър за вас
Хватко

КНИГОПИС

КНИГИ ОТ БЪЛГАРСКИ АВТОРИ

* Войников, Н. Системно програмиране за Правец-16. С., Техника, 1988, 256 с. (Поредица „16-разредни персонални компютри“)

КНИГИ ОТ СЪВЕТСКИ АВТОРИ

* Бурман, З. И., Г. А. Артюхин, Б. Я. Зархин. Программное обеспечение матричных алгоритмов и метода конечных элементов в инженерных расчетах. М., Машиностроение, 1988, 255 с.
* Бухтиаров, А. М., Ю. П. Маликова, Г. Д. Фролов. Практикум по программированию на Фортране (ОС ЕС ЭВМ). Трето изд. М., Наука, 1988, 288 с.
* Васильев, Г. П., В. Е. Горский, В. И. Шяудкулис, Н. М. Саух. Программное обеспечение неоднородных распределенных систем: анализ и реализация. М., Финансы и статистика, 1986, 160 с.
* Вишняков, Ю. С., А. И. Грюнталь, А. А. Кольцова., С. А. Пачков, А. Л. Семенов. Простое и сложное в программировании. М., Наука, 1988, 174 с. (Серия „Кибернетика — неограниченные возможности и возможные ограничения“ на АН на СССР)
* Докукина, Т. К. Программирование и алгоритмические языки. М., Машиностроение, 1988, 494 с.
* Жеребин, В. М., А. Н. Романов, Б. Е. Одинцов. Автоматизация проектирования экономических информационных систем. М., Наука, 1988, 176 с.
* Иванов, Ю. Н. Теория информационных объектов и системы управления базами данных. М., Наука, 1988, 232 с.
* Маркелова, Л. Н. Эксплуатация программоуправляемой вычислительной машины „Искра 226“. М., Машиностроение, 1987, 220 с.
* Микро-ЭВМ. В 8-мн кн. Кн. 7. Ю. И. Волков, В. Л. Горбунов, Д. И. Панфилов, С. Г. Шаронин. Учебные стенды. М., Высшая школа, 1988, 224 с.
* Осадчиев, А. А. Персональный компьютер. М., Московский рабочий, 1988, 161 с. (Библиотека делового человека)
* Поснова, М. Ф., Н. Н. Поснов. ЭВМ для всех: практическое пособие по работе с программируемыми микрокалькуляторами. Минск, Изд. Университетское, 1988, 193 с. (Университет — школе).
* Савельев, А. Я. Прикладная теория цифровых автоматов. М., Высшая школа, 1987, 272 с.
* Успенский, В. А. Машина Поста. Второ изд. М., Наука, 1988, 96 с. (Популярные лекции по математике, вып. 54)
* Хорошевский, В. Г. Инженерный анализ функционирования вычислительных машин и систем. М., Радио и связь, 1987, 256 с.
* Чехлов, Н. И., В. Ф. Игнатов, И. В. Липсиц. Цена на экране компьютера. М., Экономика, 1988, 112 с.
* Шелест, В. Д. Практикум по программированию на языке ПЛ/1. Л., Изд. ЛУ, 1986, 224 с.

ЗАДОЧЕН КОНКУРС ПО ИНФОРМАТИКА

Задача 11

от 1989 година

Даден е списък от правоъгълници в равнината, зададени с координатите на върховете си. Страните им са успоредни на координатните оси. Да се състави програма, която отпечатва координатите на точка, принадлежаща едновременно на всичките правоъгълници, ако такава съществува.

Съставил
ЕМИЛ КЕЛЕВЕДЖИЕВ

●

Задача 12

от 1989 година

Даден е списък от многоъгълници в равнината, зададени с координатите на върховете си. Въображаем плотер разполага със следните команди:

init — поставя перото в точката (0,0);

pendown — сваля перото;

penup — вдига перото;

turn a — установява посока за движение под ъгъл a;

forward n — придвижва перото на разстояние n.

Да се състави програма, която отпечатва редица от команди, необходими за начертаването на дадените многоъгълници.

Съставил
ЕМИЛ КЕЛЕВЕДЖИЕВ

Задача 1

от 1990 година

Една фирма пуска за продажба 1990 акции, като първата е със стойност 1 лев, втората — 2 лева, и т. н., последната — 1990 лева. Намират се n купувачи, които заявяват, че ще купят веднага всичките акции, ако фирмата им ги разпредели така, че всеки един от тях да получи по равен брой акции и да плати по равно. Съставете компютърна програма, която да помогне на фирмата.

Съставила
ЙОРДАНКА ГОРЧЕВА

●

Задача 2

от 1990 година

Двама играят следната игра. Дълго парче въже се разрязва последователно с ножица от първия и от втория играч. Губи този, който пръв отреже парче, по-късо от един сантиметър. Да се състави програма за компютър, която, започвайки първа, да играе срещу втория играч и да спечали играта.

Съставил
ЕМИЛ КЕЛЕВЕДЖИЕВ

●

Клубният съвет на ТНТМ към Института по математика при БАН и факултета по математика и информатика на Софийския университет заедно с редакцията на списание „Компютър за вас" продължава да провежда конкурс по информатика и през 1990 година. Конкурсът е предназначен предимно за ученици и студенти.

През тази година на страниците на списанието ще бъдат публикувани общо шест задачи. Възможно е някои от тях да са формулирани така, че да допускат уточняване или обобщаване при решаването.

Решението трябва да съдържа текста на програмата, обосновката на алгоритъма и описание на начина за работа с

нея. Ограничение на езика за програмиране няма. На тези, които се колебаят, препоръчваме Паскал. Желателно е програмата да е записана на дискета за микрокомпютър Правец—16, Правец—82 или съвместимите с тях. При оценяване на решенията освен правилността им ще се вземат предвид тяхната ефективност и евентуални идеи за обобщения.

Решението трябва да се изпрати в отделен плик, като се посочват пълни данни за участника — трите имена, училище (ВУЗ), клас (курс) и точен адрес за кореспондиране (по възможност и телефонен номер). За да бъде върната по пощата, дискетата трябва да е подходящо опакована и на отделен плик да е написан само вашият адрес.

Срокът за изпращане на решенията е два месеца след излизането на списанието. Адресът е:

1113 София
ул. Акад. Г. Бончев, бл. 8
Институт по математика с ИЦ при БАН
КС за ТНТМ
За конкурса по информатика

За всяка задача се публикува отделно класиране на участниците по точки и се раздават награди на обща стойност 100 лева.

Ще има и годишно класиране със специални награди, които ще бъдат обявени допълнително.

Най-добрите решения на участниците ще бъдат публикувани в списанието.

Очакваме на посочения адрес и писма с коментари и забележки по вече публикувани решения на задачи от конкурса, както и оригинални задачи или оргинални идеи за интересни фрагменти от програми.

Решение • Решение • Решение • Решение • Решение • Решение • Решение

Задача I от 1989 година

Да припомним условието на задача 1 от 1989 година.

Даден е набор от записи, като всички записи се състоят от еднакъв брой полета, означени с главните латински букви A, B, C, . . . и съдържат цели числа от интервала между 0 и 99. В полето „A" е зададен поредният номер на записа. Някои от имената на полетата са отбелязани. Съставете програма, която да групира записите, така че във всяка група записите да имат еднакви стойности на отбелязаните полета, а при различните групи – различни стойности. Програмата трябва да отпечатва броя на намерените групи и за всяка група – стойностите на отбелязаните полета и списък на поредните номера на записите.

Задачата не се оказа трудна за участниците. С малки изключения всички, които са изпратили решения, са я решили правилно. Затова при оценяването силно влияние оказа оформянето на програмите и удобството за работа с тях. Смисълът на условието беше разбран по два различни начина от различните участници, което е в съгласие с постановката на конкурса за даване на не толкова строго дефинирани задачи, колкото теми за разработване. Една част от участниците са сметнали, че отбелязаните полета от условието на задачата са различни при всеки запис, а останалите участници са приели, че тези отбелязани полета са едни и същи за всичките записи. Това не е оказало влияние при оценяването. Интересен е и подходът на участниците при избиране на начин за представяне на данните. Срещат се две основни идеи: използване на масиви и използване на свързани списъци с указатели. От общо 13 души изпратили решения, 11 са програмирали на Турбо Паскал, а двама на Бейсик. Един е работил на персонален компютър Правец-8Д, а останалите на компютри, съвместими с Правец-16.

Присъждат се две първи и две втори награди. Първа награда, която носи 9 точки (от 10 възможни), се дава на Емил Джалев от Благоевград и Тодор Митев от Варна. Всеки един от тях получава и парична награда от 30 лева. Втора награда (8 точки) и по 20 лева получават Владимир Алексиев от София и Илия Косев от Пловдив. Останалите участници в конкурса получават поощрителни точки, които важат за крайното класиране: Атанас Колев от Варна, Васко Томанов от Хасково, Георги Билчев от Русе, Димитър Стайков от Ямбол, Павел Бойчев от Стара Загора и Петър Стоянов от Варна – 7 т., Георги Георгиев от София и Свилен Цанов от Силистра – 6 т., и Огнян Димитров от Дряново – 5 т.

В предложената програма, написана на Турбо Паскал (версии 3, 4 или 5), основните данни се съхраняват в двумерния масив data, като първият му индекс отбелязва номера на записа в дадения набор, а вторият индекс – името на съответното поле. Въведено е едно допълнително поле с име @, в което по време на работата на програмата се записва номерът на групата, към която принадлежи съответният запис. Общият брой на групите се получава в глобалната променлива g. С цел опростяване броят на въведените записи – max, както и името на последното поле – last, са дадени в програмата като константи. В глобалната променлива flag, която е от тип множество, се записват имената

на отбелязаните полета. Данните в предложената програма се въвеждат чрез процедурата init, която осъществява зареждане на полетата на записите със случайни числа 0, 1 или 2 (например). Тя лесно може да се видоизмени, така че данните да се въвеждат от клавиатурата или по друг начин. Основната част от работата на програмата се извършва от процедурата work. Тя търси запис, който не е прикрепен към някоя група, т. е. полето @ има стойност 0. След като такъв запис бъде намерен, процедурата makeGroup увеличава брояча g с единица и записва номера на групата в полето @ на всички останали записи от тази група. Дали два записа с номера i и j са от една и съща група, се дава чрез функцията same. Накрая процедурата displayGroups служи за отпечатване на вече намерените групи в зависимост от стойността в полето с име @ .

ЕМИЛ КЕЛЕВЕДЖИЕВ

```
program zad_1_89;

const
 max=99; last='Z';

var
 data : array[0..max,'@'..last] of 0..max;
 flag : set of 'A'..last;
 g : integer;

procedure init;
 var i : integer; s : char;
begin

 {  Тази процедура дава само примерно зареждане      }
 {  на данните. Читателят може да я замени с         }
 {  друга подходяща процедура за вход на програмата. }

 for i:=0 to max do
  begin
   data[i,'@']:=0; data[i,'A']:=i;
   for s:='B' to last do data[i,s]:=random(3);
  end;
 flag:=['B','C','D','E'];
end{init};

function same(i,j:integer):boolean;
 var s : char;
begin
  for s:='A' to last do
   if s in flag then
    if data[i,s]<>data[j,s] then
     begin
      same:=false;
      exit
     end;
  same:=true;
end{same};

procedure work;
 var
  i:integer;

 procedure makeGroup;
  var j:integer;
 begin
  g:=g+1;
  data[i,'@']:=g;
  for j:=i+1 to max do
   if same(i,j) then data[j,'@']:=g;
 end{makeGroup};

begin{work}
 g:=0;
 repeat
  i:=0;
  while (data[i,'@']<>0) and (i<max) do i:=i+1;
  if data[i,'@']=0 then makeGroup;
  until i=max;
 end{work};

procedure displayGroups;
 var i,j : integer;
     s : char;
     first : boolean;
begin
 writeln('Брой на групите = ',g);
 for j:=1 to g do
  begin
   write('Група номер ',j:2,' : ');
   first:=true;
   for i:=0 to max do
    if data[i,'@']=j then
     begin
      if first then
       begin
        for s:='A' to last do
         if s in flag then
          write(s,'=',data[i,s]:2,' ');
        first:=false;
        write(' : ');
       end;
      write(data[i,'A']:2,' ');
     end;
   writeln;
  end;
end{displayGroups};

begin
 init;
 work;
 displayGroups;
end.
```

Задача II от 1989 година

Условието на втората задача от конкурса за 1989 година беше следното:

Даден е списък L от k на брой съвкупности. Всяка съвкупност е наредена n-орка от цифри: a=(a(1), a(2),...,a(n)), (a(i)=0,...,9).

Нека a е такава n-орка, че съществува индекс j такъв, че a(i)=0 за всяко i>j. Тогава n-орката b наричаме наследник на a, ако b(i)=a(i) за всяко i<=j. Да се състави програма, която отпечатва елементите на L само по веднъж, подредени така, че вдясно от всеки елемент да се намират неговите наследници (ако има такива), които пък да са подредени във вертикална колонка. Например списъкът от четворки:

1000, 1100, 1110, 1120, 1111, 1200, 1300, 1310, 1320 трябва да се отпечата като:

```
1000  1100  1110  1111
            1120
      1200
      1300  1310
            1320
```

По същество това е задача за построяване на дървовидна структура от данни. Такива структури се дефинират, построяват и обработват най-елегантно по рекурсивен начин. Достатъчно е да определим как при вече построен (или обработен) един възел от дървото, трябва да построим (или обработим) неговите наследници. При нашата задача, щом сме построили възел от вида (a(1),...,a(j),0,...,0), то трябва да построим неговите наследници, които намираме от дадения списък от n-орки. При тази задача също се появява малка неоднозначност: може да се появи непряк наследник. Например, ако в приложения към условието на задачата списък от четворки добавим още четворката 1201, тъй като няма четворка от вида 12×0, $\times <> 0$, трябва да се уточни дали подреждането трябва да изглежда така:

```
1000  1100  1110  1111
            1120
      1200  1201
      1300  1310
            1320
```

или така:

```
1000  1100  1110  1111
            1120
      1200        1201
      1300  1310
            1320
```

В получените решения на участниците в конкурса се срещат и двата варианта. Както при първата задача, начинът на доопределяне на условието не влияе при оценяването на работите. Въпреки че изглежда естествено да бъде избран такъв език за програмиране, който допуска рекурсия, някои от участниците са направили доста хубави програми на Бейсик.

Дават се две първи награди от 10 точки и по 30 лева на Владимир Алексиев от София и Тодор Митев от Варна. Втора награда от 9 точки и по 20 лева получават също двама участници: Димитър Стайков от Ямбол и Илия Косев от Пловдив. Точките, които получават всички останали участници, са следните: Красимир Генов от Плевен, Мирослав Велев от Русе и Павел Бойчев от Стара Загора – 8 т., Георги Билчев от Русе, Георги Георгиев от София и Радослав Радев от с. Врани кон, общ. Омуртаг – 7 т., Атанас Колев от Варна и Ивелин Стоянов от Ямбол – 6 т., Мария Берова от София, Петър Стоянов от Варна и Свилен Цанов от Силистра – 4 т., Емил Джалев от Благоевград – 2 т.

В предложената програма, написана на Турбо Паскал (версии от 3 до 5) с цел опростяване данните, които характеризират броя на цифрите във всяка съвкупност n и максималния брой на съвкупностите maxdata, са дефинирани като константи. Самите съвкупности от цифри се въвеждат чрез процедурата input от текстов файл с име 'data', чийто примерен вид е даден след текста на програмата. В променливата k се съдържа броят на въведените съвкупности.

Процедурата input създава също и една допълнителна съвкупност, съставена само от нули. Въведените съвкупности от цифри се пазят в масива data, чиито елементи са от тип node. Този тип е запис от две компоненти.

В първата се съдържа съвкупността от цифри като знаков низ, а втората е масив от 9 на брой числа, които по време на работа на програмата (чрез процедурата heredity) се зареждат с номерата на възможните най-много 9 наследници, намиращи се в масива data. Процедурата heredity с входен параметър j зарежда споменатите номера на наследниците на съвкупността с номер j в полетата data[j].h[1],...,data[j].h[9]. Това става, като се определи броят i на цифрите на data[j].a до първата нула и след това се обходят всички останали елементи на масива data и се отделят тези, на които първите i на брой цифри са еднакви със съответните цифри на data[j]. След това от така отделените елементи на масива data чрез процедурата make се разглеждат само тези, на които последните i+2 цифри са нули и техните номера се записват като наследници на съвкупността с номер j. След като за всяка съвкупност се създаде масив от номерата на наследниците ѝ, не е особена трудност съвкупностите да се отпечатат по искания начин. Това става чрез процедурата show, която



извиква себе си рекурсивно. Да отбележим, че в предложената програма съвкупността 1201 от видоизменения тестов пример изобщо няма да се отпечата и освен това винаги като първа ще се отпечатва съвкупността, съставена от n на брой нули.

ЕМИЛ КЕЛЕВЕДЖИЕВ

```
program zad_2_89;
const
 maxdata = 100;
 n = 4;

type
 node = record
         a : string[n];
         h : array[1..9] of 0..maxdata;
        end;

var
 data : array[0..maxdata] of node;
 level,k : integer;
 nl : boolean;

procedure input;
 var f:text;
     i : integer;
begin
 assign(f,'data');
 reset(f);
 k:=0;
 while not eof(f) do
  begin
   k:=k+1;
   readln(f,data[k].a);
   for i:=1 to 9 do data[k].h[i]:=0;
  end;
 close(f);
 data[0].a:='0';
 for i:=2 to n do data[0].a:=data[0].a+'0';
 for i:=1 to 9 do data[0].h[i]:=0;
end{input};

procedure heredity(j:integer);
 var i,m : integer;

  procedure make;
   var p,r,code : integer;
  begin
   for p:=i+2 to n do if data[m].a[p]<>'0' then exit;
   val(data[m].a[i+1],r,code);
   data[j].h[r]:=m;
  end{make};

 begin
  i:=pos('0',data[j].a)-1;
  if i>=0 then
   for m:=1 to k do
    if m<>j then
     if copy(data[j].a,1,i)=copy(data[m].a,1,i) then make;
 end{heredity};

 procedure show(j:integer);
  const  space=' ';
  var i : integer;
 begin
    if nl then
     begin
      writeln;
      for i:=1 to level-1 do write(space:n+1);
      nl:=false;
     end;
    level:=level+1;
    write(data[j].a,space);
    for i:=1 to 9 do
     if data[j].h[i]<>0 then show(data[j].h[i]);
    level:=level-1;
    nl:=true;
   end{show};

  var j : integer;

 begin
  writeln;
  input;
  for j:=0 to k do heredity(j);
  level:=1;
  nl:=false;
  show(0);
 end.
```

Файл data

```
1201
1000
1100
1110
1120
1111
1200
1300
1310
1320
```

КС 2

ПРЕБРОЯВАНЕ НА БИТОВЕТЕ

Задача

Да се състави функция на езика Си, която да приема като аргумент цяло число без знак и да връща като резултат броя на битовете в числото, които са установени в единица.

ШЕСТ БАЙТА ХЕКСПРИНТ

АВГУСТ СТОЯНОВ

Решение на КС 1

Нека разгледаме по-подробно условието. Какво означава шестнайсетична цифра в аскикод? Това са знаковете 0, 1, ... 9, A, B, ... F, които имат кодове 30, 31, ... 39, 41, 42, ... 46 (шестнайсетично). За да получим от число между 00 и 09 число между 30 и 39, очевидно трябва да добавим 30. Ако изходното число надхвърля 09, трябва към старшата и младшата тетрада да се добави още една единица и след това младшата да се вземе по модул 10 (десетично). Тук вече започва „да мирише" на десетична аритметика. Окончателно: към изходното число трябва да се добави с команда за десетично събиране 30 (шестнайсетично), а в случай че то е било по-голямо от 09, освен това да му се добави 1. На процесор 6502 десетично събиране се извършва със същата команда, с каквато и двоично, като преди това трябва да е вдигнат флагът за десетична аритметика. Тъй като командата за събиране бездруго винаги добавя и флага за пренос, логично е той да се използва за добавянето на допълнителната единица за числата над 09. За щастие командата за сравнение на акумулатора с константа 0A поставя флага за пренос точно в необходимото за последващото десетично събиране състояние, а именно 0, ако числото е под 0A и 1, ако е 0A, или повече. С това решението е завършено и изглежда така:

```
SED             ; 1 байт, 2 такта
CMP #$0A        ; 2 байта, 2 такта
ADC #$30        ; 2 байта, 3 такта (а не два, тъй като
                  флагът D е 1)
CLD             ; 1 байт, 2 такта
                ---------------
                общо 6 байта и 9 такта
```

Особено изгодно е използването на това решение на 6502, когато то се прилага за няколко цифри и в цикъла за тяхното подготвяне (прочитане от паметта) и извеждане (да кажем на печат) не се използват командите ADC и SBC (единствените на 6502, които се влияят от флага D), защото тогава SED и CLD могат да се изнесат извън цикъла и всъщност преобразуването в аскикод ще става само с двете средни инструкции. Това вече несъмнено е най-доброто възможно решение – и по време, и по памет (4 байта и 5 такта на знак!). Дори използването на таблица няма да го превъзхожда (и без да броим дължината на таблицата, ще трябват 4 байта и 6 такта!). Нека видим сега как стои въпросът при някои други популярни микропроцесори, като фамилиите 6800, 8080, Z80, 8086. При всички тях десетичното събиране става, като след команда за двоично събиране се изпълни командата за десетична корекция DAA. За беля обаче при тези процесори след команда за сравнение флагът за пренос е точно обратен на този при 6502. При фамилиите на 8080, Z80 и 8086 има команда за инвертиране на флага за пренос. Ето как изглежда решението на задачата при тях:

8080, Z80

```
CMP #0Ah
CMC
ADC #30h
DAA
```

8086

```
CMP AL, 0Ah
CMC
ADC AL, 30h
DAA -
```

Тактовете на тези процесори имат по-различен смисъл от 6502 и затова пряко сравнение на времето за изпълнение не е възможно, но въпреки това може да се каже, че тези инструкции са от най-бързите за съответните процесори, а дължината остава удивително същата – 6 байта! Този подход не върви само при фамилията на 6800 – там пренасът не може да се инвертира с една инструкция. Тук трябва да се помисли не може ли да се използва десетичната корекция и за самото сравнение с 0A. Оказва се, че може, защото командата DAA установява флага за пренос, когато той се появи от старшата тетрада след самата корекция, т. е. тя се окаже над 9. За целта числото се събира с 90 (шестнайсетично) и му се прави десетична корекция. Ако то е било над 09, флагът за пренос ще се установи в 1, старшата тетрада ще се нулира, а младшата ще се вземе по модул 10 (десетично). Остава да се добави 40 (шестн.) заедно с преноса и отново да се направи десетична корекция, която, в случай че изходното число не е било над 09, да вземе старшата тетрада (тя е станала 13) по модул 10. Ето и цялата програма:

6800

```
ADDA #$90
DAA
ADCA #$40
DAA
```

Отново дължината е 6 байта! Този начин може да се прилага и на изброените по-горе процесори с изключение на 6502, където програмата ще се удължи с един байт, защото 6502 няма инструкция за събиране без пренос и той трябва да се изчисти преди първото събиране. Не съм измислил сам алгоритъма с десетична корекция вместо сравнение, той е приложен в известната програма Дисасемблер ASMGEN за IBM PC.

В заключение: използвайте тези алгоритми винаги, когато е възможно! Те ще ви спестят много памет и машинно време!

КНИГОПИС

КНИГИ ОТ БЪЛГАРСКИ АВТОРИ

* **Димитров, Д. Елементи от информатиката. Телевизионен курс. С., Народна младеж, 1988, 123 с.**
* **Желев, Г. Ж. Видеокомпютърни системи в образованието. С., Техника, 1988, 169 с.**

КНИГИ ОТ СЪВЕТСКИ АВТОРИ

* **Петровский, А. А. Методы и микропроцессорные средства обработки широкополосных и быстропротекающих процессов в реальном времени. Минск, Наука и техника, 1988, 272 с.**
* **Программное обеспечение микро-ЭВМ. В 11-ти кн. Кн. 9. П. А. Тимофеев, В. С. Дубровин, В. С. Петровский. М., Высшая школа, 1988, 128 с.**
* **Тихомиров, М., И. Давидов. Операционная система ДЕМОС: инструментальные средства программиста. М., Финансы и статистика, 1988, 208 с.**
* **Финк, Л. М. Папа, мама, я и микрокалькулятор. М., Радио и связь, 1988, 272 с.**
* **Генкин, В. Л., И. Л. Ерош, Э. С. Москалев. Системы распознавания автоматизированных производств. Л., Машиностроение, ЛО, 1988, 246 с.**
* **Иванищев, В. В. Автоматизация моделирования потоковых систем. Л., Наука, ЛО, 1986, 143 с.**
* **Информатика и компьютерная грамотность. Сб. статии. М., Наука, 1988, 239 с.**
* **Пирогов, В. В., Н. А. Смирнов, С. Ф. Гайстеров. Сетевые средства логического проектирования абонентских систем на базе микро-ЭВМ семейства „Электроника НЦ". Рига, Зинатне, 1986, 158 с.**

КНИГИ, ПРЕВЕДЕНИ НА РУСКИ

* **Бахманн, П., М. Френцель, К. Ханцшманн и др. Программные системы. М., Мир, 1988, 288 с.**
* **Исида, Х. Программирование для микрокомпьютеров. М., Мир, 1988, 224 с. (Шести том на 11-томната японска серия по микроелектроника).**
* **Мотоока, Т., Х. Хорикоси, М. Сакаути и др. Компьютеры на СБИС. В 2-х кн. Кн. 2. М., Мир, 1988, 336 с. (Девети том на 11-томната японска серия по микроелектроника).**
* **Трауб, Дж. Г. Васильковский, Х. Вожьняковский. Информация, неопределенность, сложность. М., Мир, 1988, 184 с.**
* **Хасэгава, Х. Мир компьютеров. В 2-х кн. Прев. от япон. М., Мир, 1988. Кн. 1, 120 с. Кн. 2, 271 с.**
* **Пом, А., О. Агравал. Быстродействующие системы памяти. Прев. от англ. М., Мир, 1987, 264 с.**
* **Хоар, Ч. Взаимодействующие последовательные процессы. Прев. от англ. М., Мир, 1989, 264 с.**

КНИГИ, СВЪРЗАНИ С ИНФОРМАТИКАТА

* **Бесекерский, В. А., Н. Б. Ефимов, С. И. Зиатдинов и др. Микропроцессорные системы автоматического управления. Л., Машиностроение, ЛО, 1988, 366 с.**
* **Вертю, Ж. Ф. Куафе. Телеуправление роботами с помощью ЭВМ. М., Мир, 1989, 199 с.**
* **Одегова, В. В. Учебный процесс и ЭВМ (Дидактические проблемы управления). Львов Изд. при ЛГУ издат. объединения „Вища школа", 1988, 176 с.**
* **Синягина, Н., Л. Йорданов, Д. Димов, И. Тенев, Л. Михов. Запомнящи устройства с гъвкави магнитни дискове. С., Техника, 1988, 120 с.**
* **Кулон, Ж. Л., Ж. К. Сабонадьер. САПР в электротехнике. Прев. от англ. М., Мир, 1988, 205 с.**
* **Автоматика и вычислительная техника. Вып. 17. Сб. статии. Минск, Вышэйшая школа, 1988, 152 с.**

■

ДВЕСТА И ДВАЙСЕТ ХИТРИНИ

ИЛИ

КАКВО МОЖЕ BAG OF TRICKS

- TRAX
- INIT
- FIXCAL
- ZAP

АНГЕЛ МИТЕВ

ВЪЗМОЖНОСТИ НА РЕЖИМА ZAP

Режимът ZAP служи за обработка на информация, която се съдържа в полето за данни на всеки сектор от диска. Тази информация се разполага в области от паметта, наречени буфери. ZAP поддържа 16 буфера с дължина 256 ($FF) байта, като първият започва от адрес 204890 ($5000). Според режима данните в един диск могат да се обработват по два начина. Дисков режим на обработка – работи се със сектори и пътечки, като обемът данни, разполаган в един буфер, отговаря на един сектор от диска. Файлов режим на обработка – работи се със записи и отместване в записа, като предварително се отваря файл, който се състои от определен брой записи, чиято дължина не е задължително равна на дължината на един сектор.

Задаването на параметри към командите може да се осъществи по три начина:

– знаков – параметрите се задават като знакове в кавички, например: „APPLE“, „HELLO“ и други;

– шестнайсетичен – параметрите се записват като шестнайсетични числа, например: FA36, 43A и други;

– десетичен – параметрите се записват като десетични числа, след които се поставят точки, например: 1214., 316189. и други.

Тези начини за интерпретация на въвежданите данни се свеждат до едно: знаковите низове и десетичните числа се превръщат в шестнайсетичен код. Текстовите параметри се превръщат, като всеки знак се преобразува в еквивалентния му аскикод, от който се получава шестнайсетичният еквивалент. Преобразуването на десетичните параметри се свежда до представянето им в шестнайсетична бройна система.

РАБОТЕН ЕКРАН НА ZAP

На снимка 1 е показан екранът на режима ZAP, на който се извежда следната информация:

Линия на състоянието е редът, разположен най-горе на екрана, където е посочено текущото състояние на ZAP.

Програмата използва следните означения:

S6, 1 – активен слот е 6, активно устройство е 1;

V$?? – текущият диск е с номер на том $??;

T$?? – текущата пътечка, с която се работи, е $??;

S$?? – текущият сектор, който подлежи на обработка, е $??;

+$?? – отместването от началото на текущия буфер е $??;

#0 – текущ буфер е 0;

* – записът върху текущия диск е забранен от ZAP;

D – указател за типа на операционната система, който поддържа текущия диск;

U – указател за вида на аскиеквивалента на буфера;

I – текущ слот с контролера на принтера.

Командна линия е редът под линията на състоянието, на който се изписват командите и параметрите към ZAP.

Колона на адресите с най-лявата колона на екрана, от която потребителят се информира за разположението на данните в буфера.

Шестнайсетично състояние на буфера е средната колона на екрана, в която в шестнайсетичен код е показано съдържанието на целия буфер.

Аскиеквивалент на буфера е най-дясната колона, в която чрез аскикод са изобразени знаковете, отговарящи на съот-

ветните стойности от буфера, като контролните символи се изобразяват с точки.

ZAP се управлява от команди с различни параметри, които се въвеждат на командната линия. За правилното въвеждане трябва да се има предвид действието на следните клавиши:

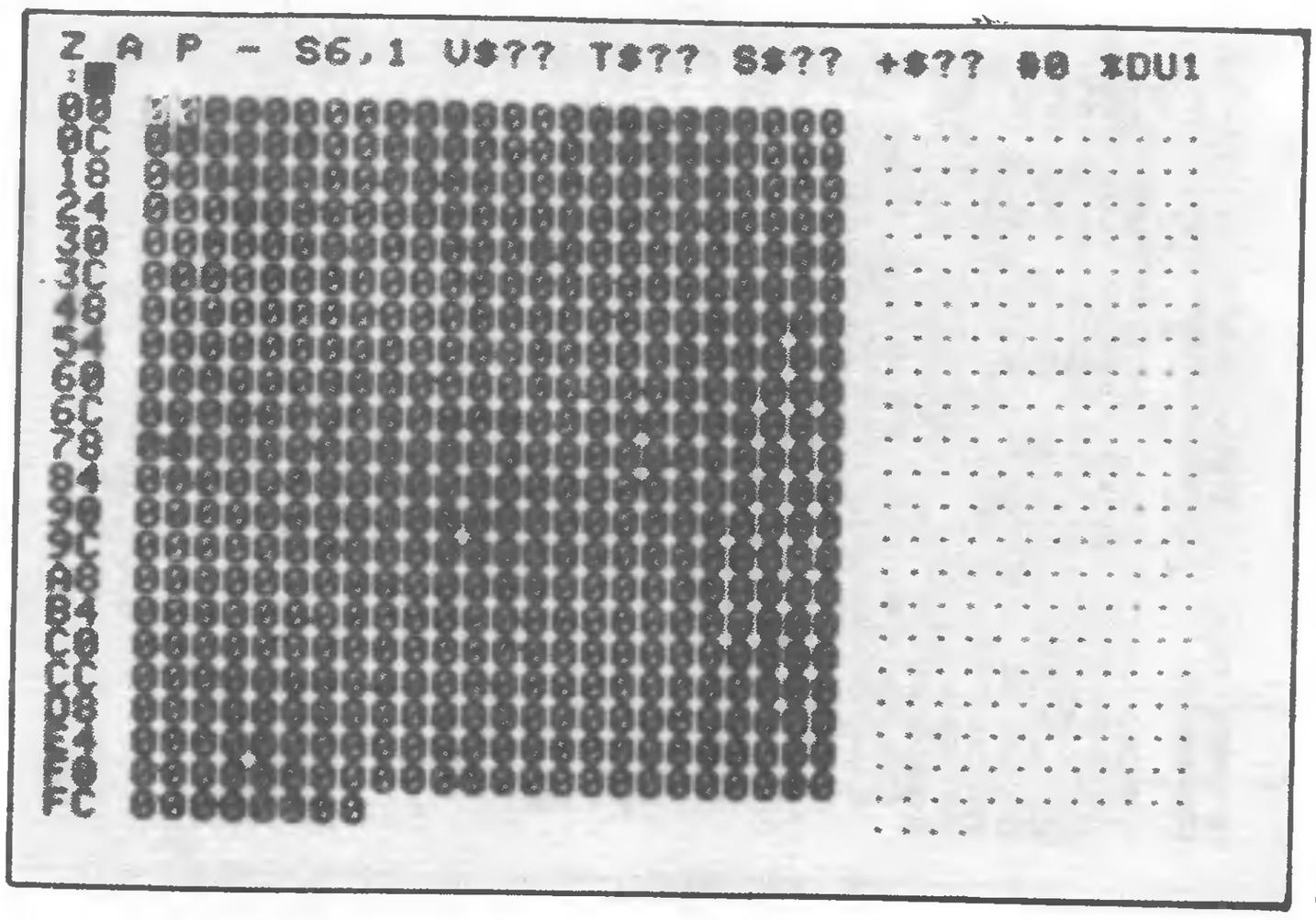
RETURN – привежда в изпълнение от ZAP написаното на командната линия;

CB+E – чисти командната линия от текущото положение на курсора до края на реда;

MK+U – прочита данни от командната линия;

MK+H – отменя данни от командната линия.

Командите и прилежащите им параметри в ZAP се въвеждат слято, защото интервалът се интерпретира като разделител между две команди.



Сн. 1

ВХОДНО—ИЗХОДНИ КОМАНДИ

израз – премества курсора на текущия буфер с отместване, чиято стойност е *израз*.

+*израз*...или...–*израз* – премества курсора на текущия буфер на ново отместване, изчислено чрез прибавяне или изваждане на *израз*.

R*пътечка*, *сектор* – в дисков режим от текущия диск се чете *сектор* от *пътечка*, като прочетеният сектор се разполага в текущия буфер. Ако отсъства някой от параметрите, по подразбиране се задават текущите.

R*запис*, *байт* – във файлов режим се чете номер на запис и се разполага на отместване в записа според зададените *запис* и *байт*, като прочетеното се разполага в текущия буфер. Ако отсъства някой от параметрите, по подразбиране се задават текущите.

WRITE*запис*, *байт* – във файлов режим текущият буфер се записва от отместване, зададено чрез *байт*, в запис с номер, зададен от *запис*. Ако някой от параметрите отсъства, по подразбиране се задават текущите.

WRITE*пътечка*, *сектор* – в дисков режим на текущия диск се записва текущият буфер върху сектор от пътечка, определени с *пътечка* и *сектор*. Ако някой от параметрите отсъства, по подразбиране се задават текущите.

N*сектор* – в дисков режим от текущия диск се чете секторът, определен от стойността на сбора от номера на текущия сектор и стойността на *сектор*, като прочетеното се разполага в текущия буфер. Ако параметърът бъде пропуснат, по подразбиране се прочита следващият сектор.

N*запис* – във файлов режим се чете запис с номер, определен от стойността на сбора от номера на текущия запис и стойността на *запис*, като прочетеното се разполага в текущия буфер. Ако параметърът бъде пропуснат, по подразбиране се прочита следващият запис.

P*сектор* – в дисков режим от текущия диск се чете секторът, определен от разликата между номера на текущия сектор и стойността на *сектор*, като прочетеното се разполага в текущия буфер. Ако параметърът бъде пропуснат, по подразбиране се чете предходният сектор.

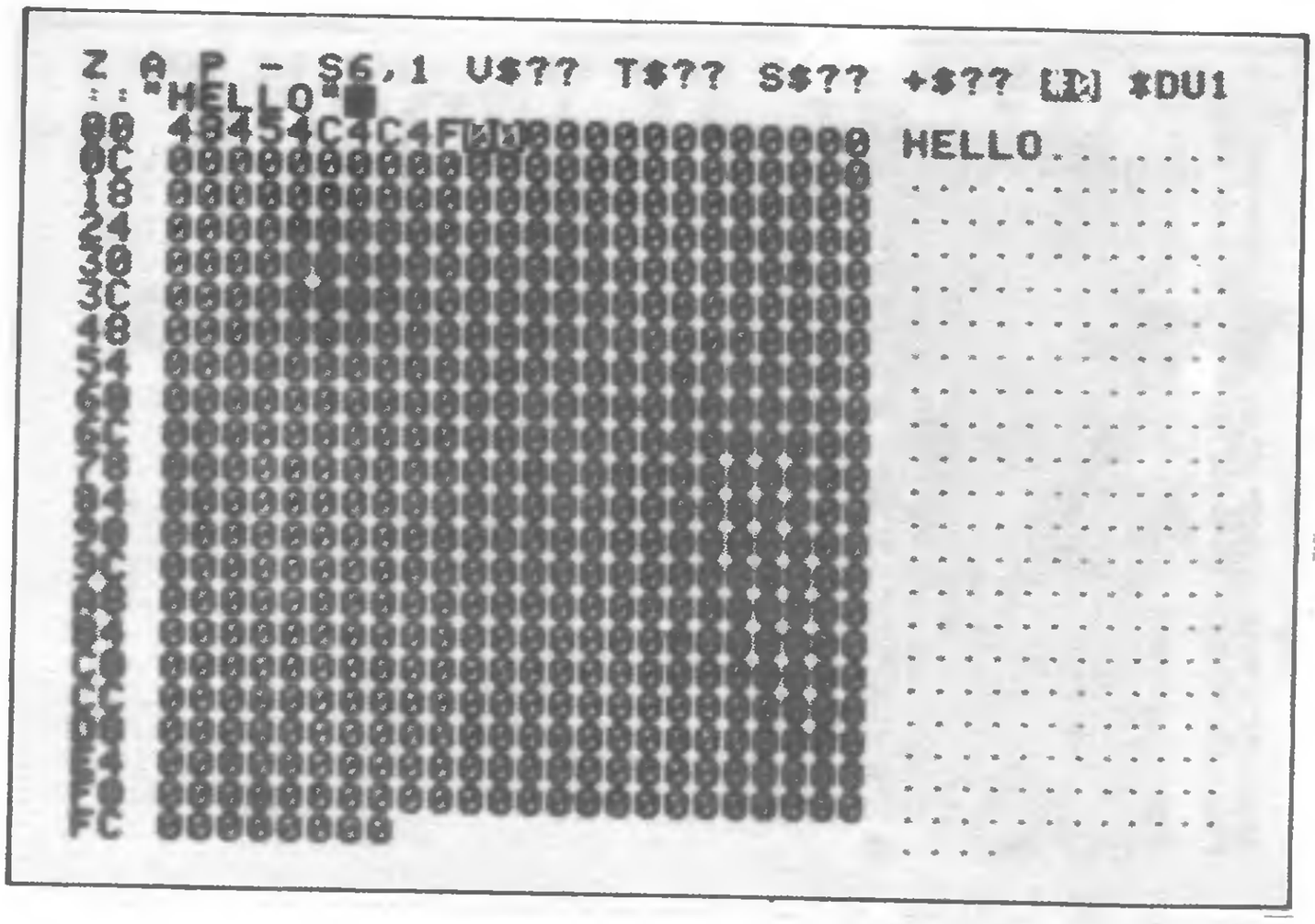
P*запис* – във файлов режим се чете запис с номер, определен от разликата между номера на текущия запис и стойността на *запис*, като прочетеното се разполага в текущия буфер. Ако параметърът бъде пропуснат, по подразбиране се чете предходният запис.

% – в дисков режим се чете от пътечка, определена от стойността на байта, върху който се намира курсорът в текущия буфер, и сектор, определен от стойността на байта, който се намира след курсора в текущия буфер, като прочетеното се разполага в текущия буфер.

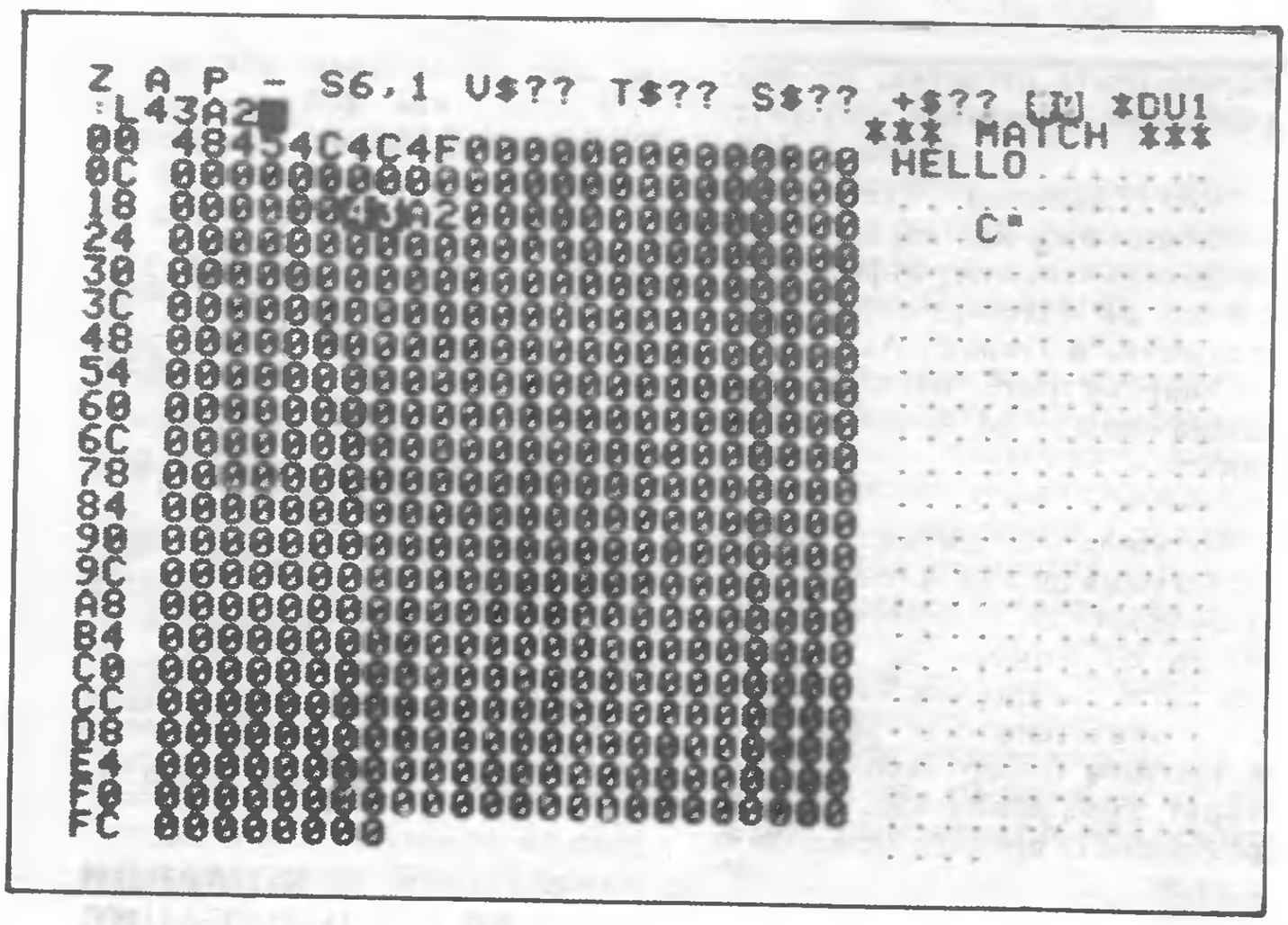
КОМАНДИ ЗА МОДИФИКАЦИЯ НА БУФЕРА

:*израз* – заменя се съдържанието на текущия буфер от текущата позиция на курсора в него с *израз* и курсорът се разполага непосредствено след заменените байтове. На снимка 2 се вижда как след написване на низа за заменяне „HELLO“, петте байта от $00 по $04 са заменени с аскикодовете на HELLO и отместването на буфера от +$00 с променено на +$05.

SET*израз* – запълва с *израз* текущия буфер от текущата



Сн. 2



Сн. 3

позиция на курсора в него.

&израз – изпълнява се логическата операция И между байтовете на *израз* и байтовете, съответни на *израз* от текущата позиция на курсора в текущия буфер. Резултатът от операцията се разполага от текущата позиция в съответните байтове.

Oизраз – изпълнява се логическата операция ИЛИ между байтовете на *израз* и байтовете, съответни на *израз* от текущата позиция на курсора в текущия буфер. Резултатът от операцията се разполага от текущата позиция в съответните байтове.

Xизраз – изпълнява се логическата операция ИЗКЛЮЧВАЩО ИЛИ между байтовете на *израз* и байтовете, съответни на *израз* от текущата позиция на курсора в текущия буфер. Резултатът от операцията се разполага от текущата позиция в съответните байтове.

КОМАНДИ ЗА СРАВНЕНИЕ

Lизраз – в дисков режим в текущия диск се търси стойност, равна на израз, като се започва от позицията след текущата позиция на курсора в текущия буфер. Във файлов режим в текущия файл се търси стойност, равна на *израз*, като се започва от позицията след текущата позиция на курсора в текущия буфер. При откриване на *израз* секторът или записът се визуализират на екрана в текущия буфер, като курсорът на буфера се позиционира върху началото на открития *израз*. Ако *израз* не бъде открит в диска или файла, картината на екрана не се променя, единствено се появява съобщение за прекратяване на търсенето. На снимка 3 се вижда как след въвеждането на израза за търсене $43A2 ZAP го открива и позиционира курсора на буфера с отместване от +$00 на +$16.

V*израз* – проверява се дали *израз* съвпада със съответните байтове под курсора в текущия буфер, като се издават съответните съобщения.

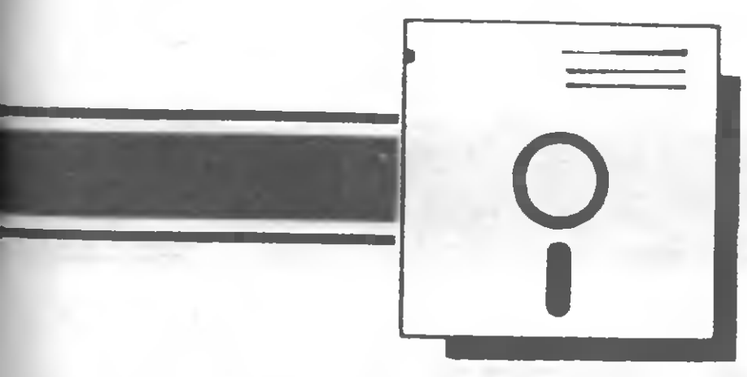
COMPARE*израз* – съдържанието на текущия буфер се сравнява с буфер, номерът на който е зададен с *израз*, от текущото положение на курсора до края. При разлика в съдържанията курсорът се позиционира върху нея и ZAP издава съобщение.

ОПЦИОННИ КЛЮЧОВЕ

LC – в колоната с аскиеквивалента на буфера малките букви и кирилицата се изобразяват съответно като малки букви и кирилица (снимка 4).

UC – в колоната с аскиеквивалента на буфера малките букви и кирилицата се изобразяват като големи латински букви (снимка 5).

IMAGE – в колоната с аскиеквивалента на буфера знаковете с шестнайсетичен код от $00 до



$3F се изобразяват в режим INVERSE, знаковете с код от $40 до $7F – във FLASH, а знаковете с код от $80 до $FF – в NORMAL (снимка 6).

ASCII – в колоната с аскиеквивалента на буфера се отменя режим IMAGE и се включва режим LC.

LOCK – забранява се записът от ZAP върху текущия диск, като в линията на състоянието се поставя „*".

UNLOCK – отменя се забраната за запис от ZAP върху текущия диск, като в линията на състоянието „*" се изтрива.

DOS13 – проверява дали текущият диск се поддържа от ДОС 3.1 или ДОС 3.2 и ако да, превключва ZAP за работа с тях.

DOS16 – проверява дали текущият диск се поддържа от ДОС 3.3 и ако да, превключва ZAP за работа с нея.

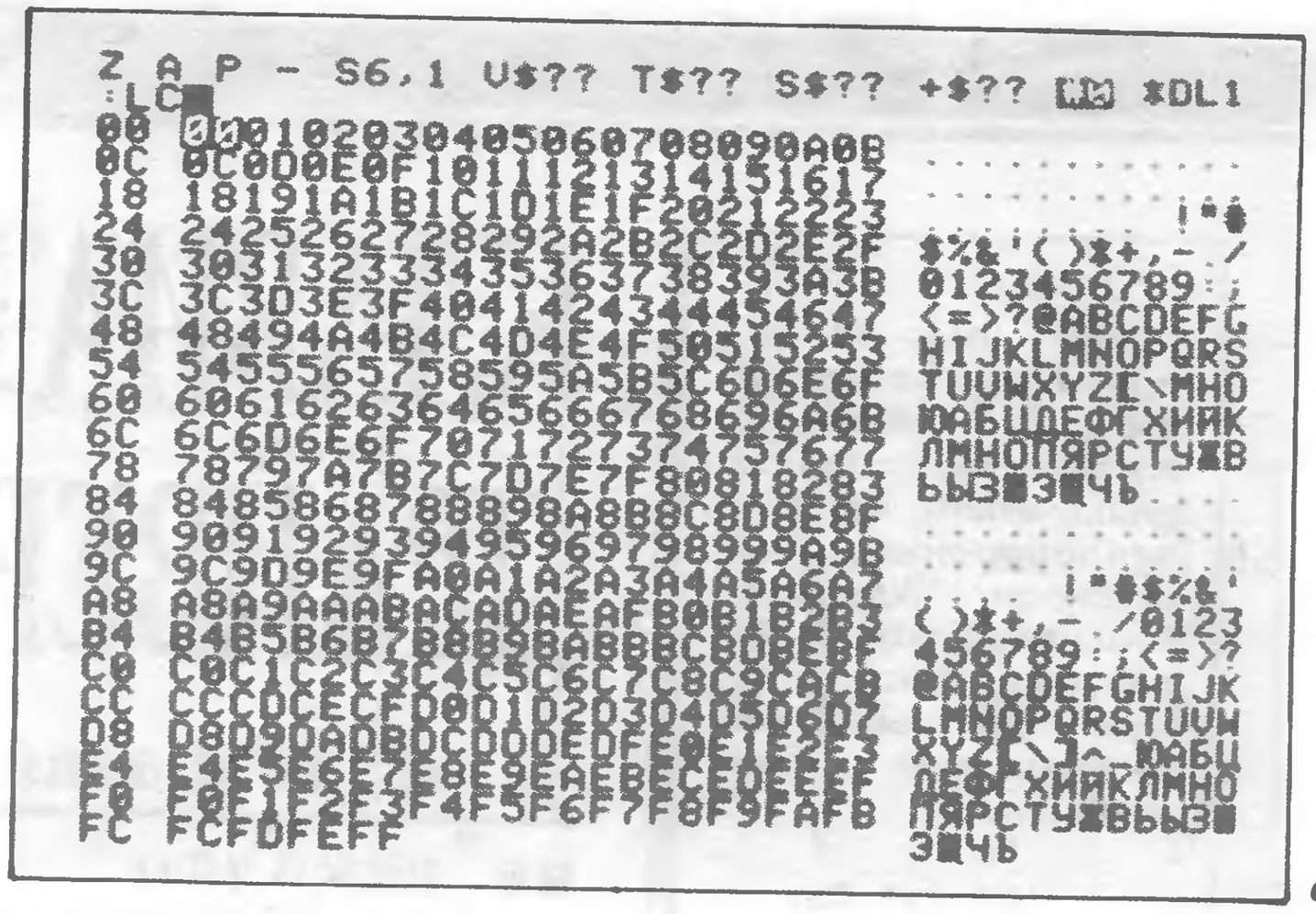
PASCAL – проверява дали текущият диск се поддържа от UCSD–PASCAL и ако да, превключва ZAP за работа с нея.

CPM – проверява дали текущият диск се поддържа от CP/M и ако да, превключва ZAP за работа с нея.

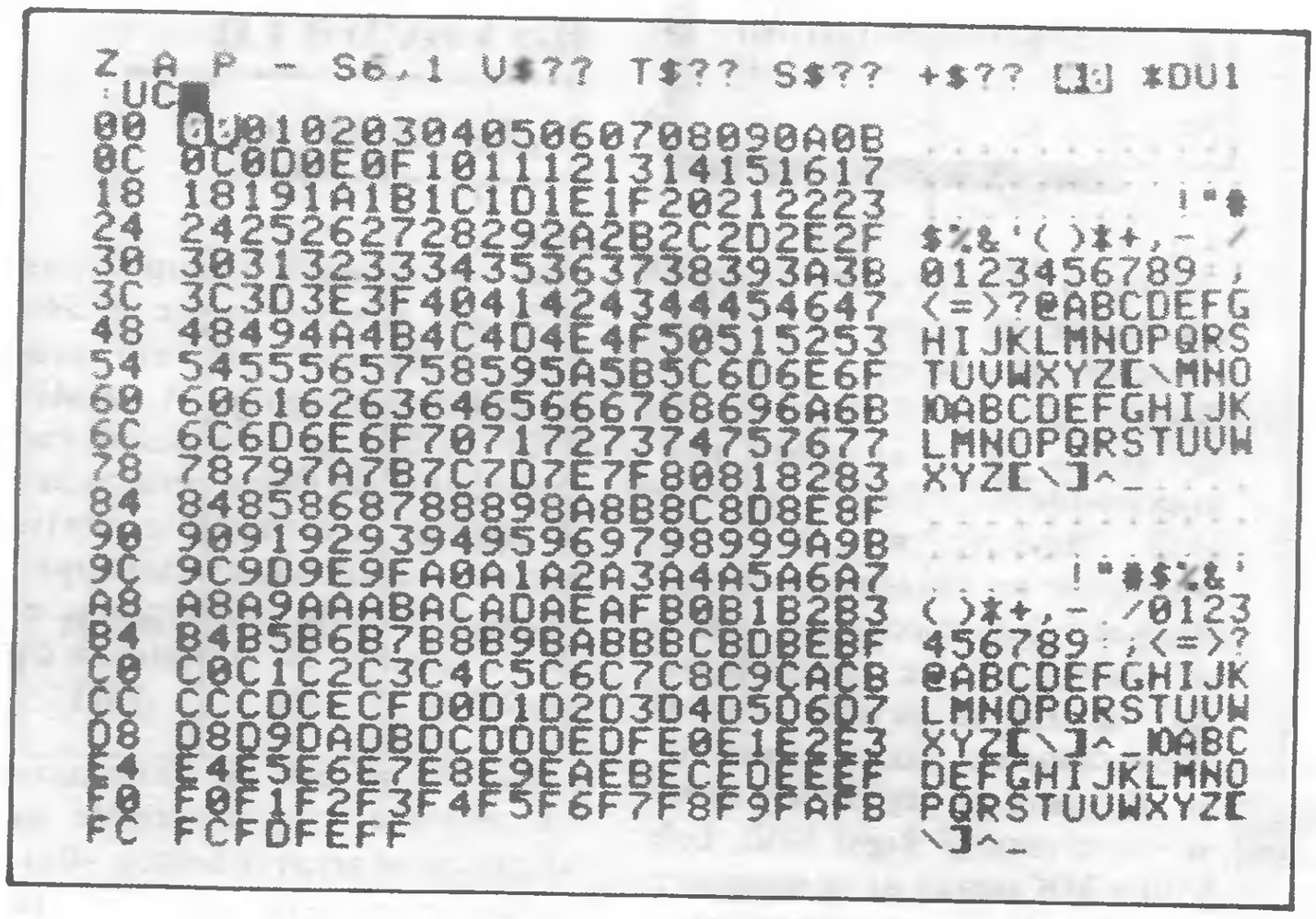
WRAP – позволява след прочитане на последния сектор в дисков режим или последния запис във файлов режим да се продължи четенето от началото на диска или файла.

NOWRAP – създава условие при прочитане на последния сектор от диска в дисков режим или последния запис на файла във файлов режим да се прекрати четенето и да се издаде съобщение.

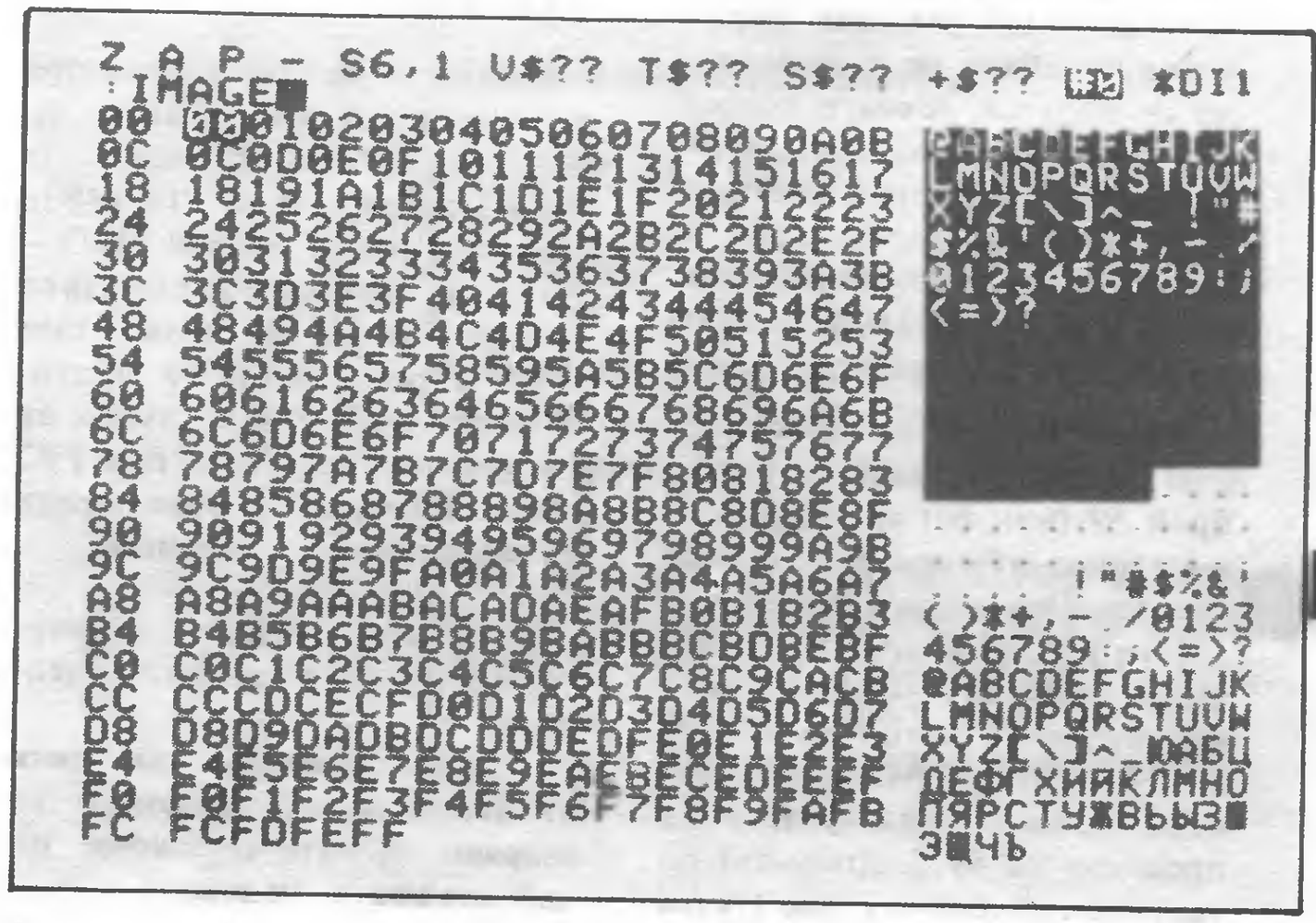
СЛЕДВА



Сн. 4



Сн. 5



Сн. 6

БОРИСЛАВ ЗАХАРИЕВ

Ние сме трима приятели, притежатели на Правец – 8Д. Имаме игри, които различават няколко едновременно натиснати клавиша. Молим да публикувате подробно решение и обяснение, как се реализира разпознаването.

Николай, Сашо и Радо,
ученици от НПМГ
— София

КЛАВИАТУРЕН ДРАЙВЕР

за разпознаване на няколко натиснати клавиша

Докато клавиатурата на Правец-82 е реализирана със специализиран контролер, тази на Правец-8Д се обслужва чрез програма. Тя се активира при прекъсване от TIMER1 на VIA 6522. Загубата на време за сканиране на клавиатурата при Правец-8Д се компенсира с възможността да се определи не само моментът на натискане на даден клавиш, но и моментът на отпускането му. Нещо повече – клавишите Right Shift, Left Shift и МК могат да се прочетат отделно. При едновременното натискане на два или повече клавиши обаче не е възможно да се получат техните кодове. По-точно казано, системната програма, която чете клавиатурата, не може да върши това. За да се определи кои клавиши са натиснати едновременно, е необходим НОВ клавиатурен драйвер.

Клавиатурата на Правец-8Д е от матричен тип. Бутоните на брой 59 (или 60) са подредени в матрица от осем реда и осем колони. Очевидно е, че не във всички пресечни точки (8*8 = 64) има поставени клавиши. Осемте колони са свързани към порт А на VIA 6522 (PA0 – PA7) през паралелния порт на музикалния процесор SY 8912. Достъпът до тези осем колони се осъществява чрез системната подпрограма W8912 с начален адрес #F590. При обръщение към тази подпрограма регистърът А на МП 6502 трябва да е зареден със стойността #E (14), а регистърът X трябва да съдържа данните, които ще се извеждат към порта на музикалния процесор, т. е. към колоните на матрицата от клавиши.

Осемте редове на матрицата са свързани към входовете на аналоговия мултиплексор 4051, който се намира на платката с клавишите.

Изходът на мултиплексора е инвертиран чрез транзистор и може да бъде прочетен на перо 3 от порт В на VIA (PB3). Пера 0, 1 и 2 от порт В (PB0 – PB2) са инициализирани като изходи. Те са свързани към трите адресни шини на мултиплексора 4051. В зависимост от съдържанието на PB0, PB1 и PB2 се адресира един от осемте реда на матрицата от клавиши.

При разглеждане на клавиатурния драйвер се вижда следното:

– **последователно към всяка от колоните на матрицата се изпраща 0, като се започне от най-младшата колона;**

– **за всяка 0 към съответната колона се адресира аналоговият мултиплексор от 0 до 7. Това съответства на последователно свързване на всеки от редовете на матрицата към изхода на мултиплексора;**

– **ако натиснатият клавиш е свързан към колоната, на която е подадена 0, и към избрания от мултиплексора ред, изходът на мултиплексора ще се установи в 0, т. е. нулата от колоната „ще премине“ през клавиша, реда и мултиплексора;**

– **поради това, че изходът на мултиплексора е инвертиран с транзистор, указание за натиснат клавиш ще бъде установяването на PB3 (инициализиран като вход) в 1;**

– **при сканирането на клавиатурата по редове и колони в буфер с начален адрес #41 се записват сканкодовете на клавишите, а в клетка #40 – техният брой.**

Сканкодът не е аскикод! Той е шестбитов (от 0 до 63) и отразява положението на клавиша в матрицата. По-точно неговите 3 младши бита са номера на реда, а трите старши бита – номера на колоната.

Внимание! Драйверът изработва безпогрешно кодовете на един и два едновременно натиснати клавиши. При три натиснати клавиша, когато те образуват прав ъгъл в матрицата, може да се получи код на четвърти клавиш, без той да е натиснат.

Това е кодът на клавиша, който образува с останалите три натиснати клавиша правоъгълник в матрицата. За щастие това е рядка комбинация, но при четири и повече едновременно натиснати клавиша това трябва да се има предвид. Създателите на игри или други програми, които ще използват драйвера, лесно могат да изберат „безконфликтни“ клавиши. А както се досещате, това са например 8-те клавиша от един ред или една колона на матрицата.

При използването на драйвера всички клавиши са с ЕДНАКЪВ приоритет. Например при натискането на клавиша N се изработва кодът 1 и PEEK (#40) = 1. При натискането на Left Shift се изработва кодът 36 и PEEK (#40) също е равно на 1. При натискането на двата клавиша обаче се изработват и двата кода – 1 и 36, а PEEK (#40) = 2. Кодовете на клавишите в десетичен вид са дадени в таблицата.

Поради разликата в клавиатурите при различните партиди Правец-8Д най-добре е да определите сканкодовете на клавишите чрез следната елементарна програма, като клавиатурният драйвер трябва да е зареден в паметта от начален адрес #400:

```
10 CLS
20 CALL #400
30 PRINT@ 2,2;PEEK(#41)
   : GOTO 20
```

Таблица

Клавиш	Код	Клавиш	Код	Клавиш	Код	Клавиш	Код
1	5	ОСВ	13	МК	20	Left Sh	36
2	22	Q	14	A	53	Z	21
3	7	W	55	S	54	X	6
4	19	E	51	D	15	C	23
5	2	R	10	F	11	V	3
6	17	T	9	G	50	B	18
7	0	Y	48	H	49	N	1
8	56	U	40	J	8	M	16
9	25	I	41	K	24	,	33
0	58	O	42	L	57	.	34
:	27	P	43	]	26	/	59
–	63	@	47	[	31	Right Sh	60
;	30	/	46	CR	61		
		^	62				
Ляво	37	Долу	38	Горе	35	Дясно	39
DEL	45	SPC	32	F1	4		


```
1   ********************************
2   *                              *
3   *    КЛАВИАТУРЕН ДРАЙВЕР       *
4   *                              *
5   *    ОТ БОРИСЛАВ ЗАХАРИЕВ      *
6   *                              *
7   ********************************
8
9           ORG  $0400      ;ДРАЙВЕРЪТ Е ПРЕМЕСТВАЕМ
10
11  MASK    =    $3E        ;ЗА ПРЕМЕСТВАЩА СЕ ВЛЯВО '0'.
12  TEMP    =    $3F        ;СТАРШИ 3 БИТА ОТ КОДА НА КЛАВИША.
13  BUFFER  =    $40        ;БРОЯ + КОДОВЕ НА КЛАВИШИТЕ.
14  PORTB   =    $300       ;ПОРТ 'B' НА VIA 6522.
15  W8912   =    $F590      ;П-НА ЗА ЗАПИС В SY 8912.
16
```

БОРИСЛАВ ЗАХАРИЕВ

КАК ДА СВИРИМ АКОРДИ

Клавиатурният драйвер за определяне на няколко едновременно натиснати клавиша разкрива нови възможности за създателите на програмни продукти за домашния компютър Правец – 8Д. Игрите ще станат по-интересни и с възможност да се играят от двама души едновременно. Любителите на компютърната музика сигурно се досещат, че клавиатурният драйвер заедно с триканалния музикален процесор SY 8912 може да превърне Правец – 8Д в триканален музикален инструмент. Съществуват различни възможности за това. Тук предлагам едно сравнително просто решение на Бейсик, чрез което могат да се изпълняват едногласни, двугласни и тригласни мелодии.

```
0400: 78        17          SEI             ;ЗАБРАНА НА ПРЕКЪСВАНИЯТА.
0401: A9 00     18          LDA  #$00       ;НУЛИРАНЕ НА БРОЯЧА
0403: 85 40     19          STA  BUFFER     ;НА АКТИВНИТЕ КЛАВИШИ.
                20
0405: A9 FE     21          LDA  #%11111110 ;БИТО 0 = 0, Т.Е. СЕ АДРЕСИРА
0407: 85 3E     22          STA  MASK       ;КОЛОНА 0 ОТ КЛАВИАТУРАТА.
                23
0409: A6 3E     24  LOOP1   LDX  MASK       ;ПРОЧИТАНЕ НА МАСКАТА.
040B: A9 0E     25          LDA  #$0E       ;ДОСТЪП ДО ПОРТА НА SY 8912
040D: 20 90 F5  26          JSR  W8912      ;ЧРЕЗ СИСТЕМНАТА П-МА W8912.
                27
0410: A2 10     28          LDX  #%00010000 ;АДРЕСИРАНЕ НА РЕДИЦА 0, А
0412: 8E 00 03  29  WPORT   STX  PORTB      ;БИТ4=1 (СТРОБ НА ПРИНТЕРА).
                30
0415: A9 08     31          LDA  #%00001000 ;ПРОВЕРКА НА БИТ3, КОЯТО
0417: 2C 00 03  32          BIT  PORTB      ;Е ИЗХОД НА MX 4051.
041A: D0 0C     33          BNE  KEY        ;ИМА НАТИСНАТ КЛАВИШ.
041C: E8        34  NEXTKEY INX             ;СЛЕДВАЩА РЕДИЦА.
041D: E0 18     35          CPX  #%00011000 ;ПРОВЕРЕНИ СА 8-ТЕ РЕДИЦИ?
041F: 90 F1     36          BCC  WPORT      ;ОЩЕ НЕ.
0421: 38        37          SEC             ;ТАКА БИТО ЩЕ СЕ УСТАНОВИ В 1
0422: 26 3E     38          ROL  MASK       ;И ЩЕ СЕ АДРЕСИРА НОВА КОЛОНА.
0424: B0 E3     39          BCS  LOOP1      ;НЕ Е ДОСИГНАТА 8-А КОЛОНА.
0426: 58        40          CLI             ;РАЗРЕШАВАНЕ НА ПРЕКЪСВАНИЯТА.
0427: 60        41          RTS             ;ИЗХОД ОТ ПРОГРАМАТА.
                42
0428: 8A        43  KEY     TXA             ;НОМЕР НА РЕДИЦАТА В РЕГ. А.
0429: 29 07     44          AND  #%00000111 ;ВАЛИДНИ СА БИТО,1 И 2.
042B: 0A        45          ASL             ;ТЕ СЕ ПРЕМЕСТВАТ
042C: 0A        46          ASL             ;НА МЯСТОТО НА
042D: 0A        47          ASL             ;БИТОВЕ 3,4 И 5.
042E: 85 3F     48          STA  TEMP       ;ВРЕМЕННО ЗАПАЗВАНЕ.
0430: A5 3E     49          LDA  MASK       ;ОПРЕДЕЛЯНЕ НА ПОЛОЖЕНИЕТО
0432: A0 00     50          LDY  #$00       ;НА АКТИВНАТА 0
0434: 4A        51  SHIFT   LSR             ;ЧРЕЗ ПОСЛЕДОВАТЕЛНИ
0435: 90 03     52          BCC  OK         ;  ПРЕМЕСТВАНИЯ НАДЯСНО.
                53
0437: C8        54          INY             ;В РЕГ. Y СА БРОЯ ПРЕМЕСТВАНИЯ
0438: B0 FA     55          BCS  SHIFT      ;НА 0-ТА, Т.Е. МЛАДШИТЕ 3 БИТА
043A: 98        56  OK      TYA             ;ОТ КОДА НА КЛАВИША.
043B: 05 3F     57          ORA  TEMP       ;ВЪЗСТАНОВЯВАНЕ НА КОДА.
                58
043D: A4 40     59          LDY  BUFFER     ;БРОЯ НА АКТИВНИТЕ КЛАВИШИ.
043F: 99 41 00  60          STA  BUFFER+1,Y ;КОДА НА ПОРЕДНИЯ КЛАВИШ.
0442: E6 40     61          INC  BUFFER     ;БРОЯ = БРОЯ+1.
0444: D0 D6     62          BNE  NEXTKEY    ;ВМЕСТО JMP.
                63
```

НА ПРАВЕЦ-8Д

Клавиатурният драйвер се зарежда в паметта чрез оператора DOKE, като машинните кодове на програмата са зададени с оператори DATA. Втората група от оператори DATA представлява сканкодове на клавиши. Тези кодове са подредени възходящо, т. е. първият код 36 (Left Shift) съответства на най-ниския тон (в случая МИ), следващият код 21 (клавиш Z) е с половин тон по-висок – ФА, и т. н. Според предпочитанията си всеки може сам да подреди сканкодовете в оператори DATA. Последният от сканкодовете е 61 и в условието UNTIL е указател за край на DATA. При нова наредба на кодовете може да се наложи да се промени условието UNTIL.

Константата M = 2^(1/12) (или казано с думи, две на степен една дванайсета) представлява съотношението между два съседни полутона в музиката. С нея се изчисляват всички тонове, като за начална стойност е избрано числото 800 – променливата NO. При смяна на началната стойност се получава транспониране, като по този начин с възможна донастройка към друг музикален инструмент.

Генерираният звук зависи от това, дали са натиснати един, два или три клавиша. Повече от три едновременно натиснати клавиша се пренебрегват. При един натиснат клавиш първият генератор на SY 8912 генерира съответния тон. Вторият генератор – същия тон, но с една октава по-ниско, а третият генератор – тон с период малко по-голям от този на първия генератор. Така се получава унисонно звученс, което прави звука по-колоритен. При два натиснати клавиша единият от тоновете е нормален, а другият – в унисон. При три натиснати клавиша всеки от генераторите генерира съответен тон. Така на Правец-8Д могат да се изпълняват акорди.

По време на изпълнение могат да се променят два параметъра на свирене – сила и отзвучаване. Лявата и дясната стрелка променят отзвучаването, а стрелките „нагоре" и „надолу" – силата на звука. И двата параметъра са визуализирани с цветни ивици и число от 0 до 15.

Програмното реализиране на ефекта „отзвучаване" е чрез цикъл, в който операторът SOUND е с намаляваща сила на звука. Възможна е по-лесна реализация на отзвучаването чрез комбинацията от операторите SOUND и PLAY. При подходящи параметри на PLAY се използва вграденият в SY 8912 генератор на обвивка. Така обаче се чува неприятно пукане (макар и тихо) при постепенното затихване на звука.

Накрая ще дам три съвета!

1. Ако за някои тази програма се окаже „бавна" и те не могат да свирят ноти с продължителност 1/32, ги съветвам да я пренапишат на асемблер, като обаче използват същия клавиатурен драйвер.

2. На Hi-Fi любителите препоръчвам да включат компютъра си към НЧ усилвател.

3. При различните партиди Правец-8Д са използвани различни видове усилватели чипове LM 386. При някои от чиповете липсва вътрешен резистор между вход и земя, което е причина за силно изкривяване на генерирания от SY 8912 звук. Добавете резистор 1...3к между масата на компютъра и извод № 3 на LM 386.

Приятни занимания с новия гигласен инструмент!

```
10 PRINT"ЗАРЕЖДАНЕ";
20 FOR I=#400 TO #445 STEP 2
30 READ D$:D$="#"+D$:D=VAL(D$)
40 DOKE I,D:PRINT".";:NEXT
50 ' ДРАЙВЕР СКАНИРАЩ КЛАВИАТУРАТА
60 DATA A978,8500,A940,85FE,A63E,A93E
70 DATA 200E,F590,10A2,8E,A903,2C08
80 DATA 300,CD0,E0E8,9018,38F1,3E26
90 DATA E3B0,6058,298A,A07,A0A,3FB5
100 DATA 3EA5,A0,904A,C803,FAB0,398
110 DATA A43F,9940,41,40E6,D6D0
120 ' КОД НА СКАНИРАНЕ - ПОРЕДЕН ТОН
130 DATA 36,21,54,6,15,23,11,3,18,49,1
140 DATA 8,16,33,57,34,26,59,31,60
150 DATA 13,5,14,22,55,51,19,10,2,9
160 DATA 17,48,40,56,41,25,42,43,27
170 DATA 47,63,46,30,62,61
180 POKE #26A,10:CLS:INK7:PAPER0
190 POKE #30E,#40 ' ЗАБРАНА НА TIMER1
200 M=2^(1/12) 'МЕЖДУТОНОВ КОЕФИЦИЕНТ
210 NO=800 'ПЕРИОД НА НАЙ-НИСКИЯ ТОН
220 S0=#40:S1=#41:S2=#42:S3=#43:X=15
230 UP=35:LE=37:DO=38:RI=39:DI=100
240 DIM N(63) 'ПЕРИОДИ НА ТОНОВЕТЕ
250 REPEAT:READ I
260 NO=NO/M:N(I)=NO+.5
270 UNTIL I=61 'ПОСЛЕДНО ЧИСЛО В DATA
280 POKE S1,0:PLAY 7,0,0,0
290 FORL=0 TO 6:PLOT L+20,0,#11:NEXT:L=L-1
300 FORX=0 TO 15:PLOT X+20,2,#14:NEXT:X=X-1
310 PLOT0,0,"ОТЗВУЧАВАНЕ":PLOT0,2,"СИЛА НА ЗВУКА"
320 PLOT L+21,0,#10:PLOT L+20,0,#11
330 PLOT 14,0,STR$(L)+" "
340 PLOT X+21,2,#10:PLOT X+20,2,#14
350 PLOT 14,2,STR$(X)+" "
360 VO=0:GOTO 380
370 VO=X 'НАЧАЛНА СИЛА НА СВИРЕНЕ
380 FOR V=VO TO 0 STEP-1
390 SOUND1,N1,V:SOUND2,N2,V:SOUND3,N3,V
400 FOR D=0 TO L:CALL#400:KN=PEEK(S0)
410 IF KN>0 THEN 430
420 NEXT:NEXT:VO=0
430 IF KN>3 THEN KN=3
440 KY=PEEK(S1):N1=N(KY)
450 ON KN+1  GOTO 380,460,510,520
460 IF KY=UP THEN X=X-(X(15):GOTO 320
470 IF KY=DO THEN X=X+(X<>0):GOTO 320
480 IF KY=LE THEN L=L+(L<>0):GOTO 320
490 IF KY=RI THEN L=L-(L(15):GOTO 320
500 N2=N1-N1/DI:N3=N1+2:GOTO 370
510 N2=N(PEEK(S2)):N3=N1-N1/DI:GOTO 370
520 N2=N(PEEK(S2)):N3=N(PEEK(S3)):GOTO 370
```

ДЖОЙСТИК

ЗА ПРАВЕЦ-8Д

Инж. ГЕОРГИ ВЛАДИМИРОВ

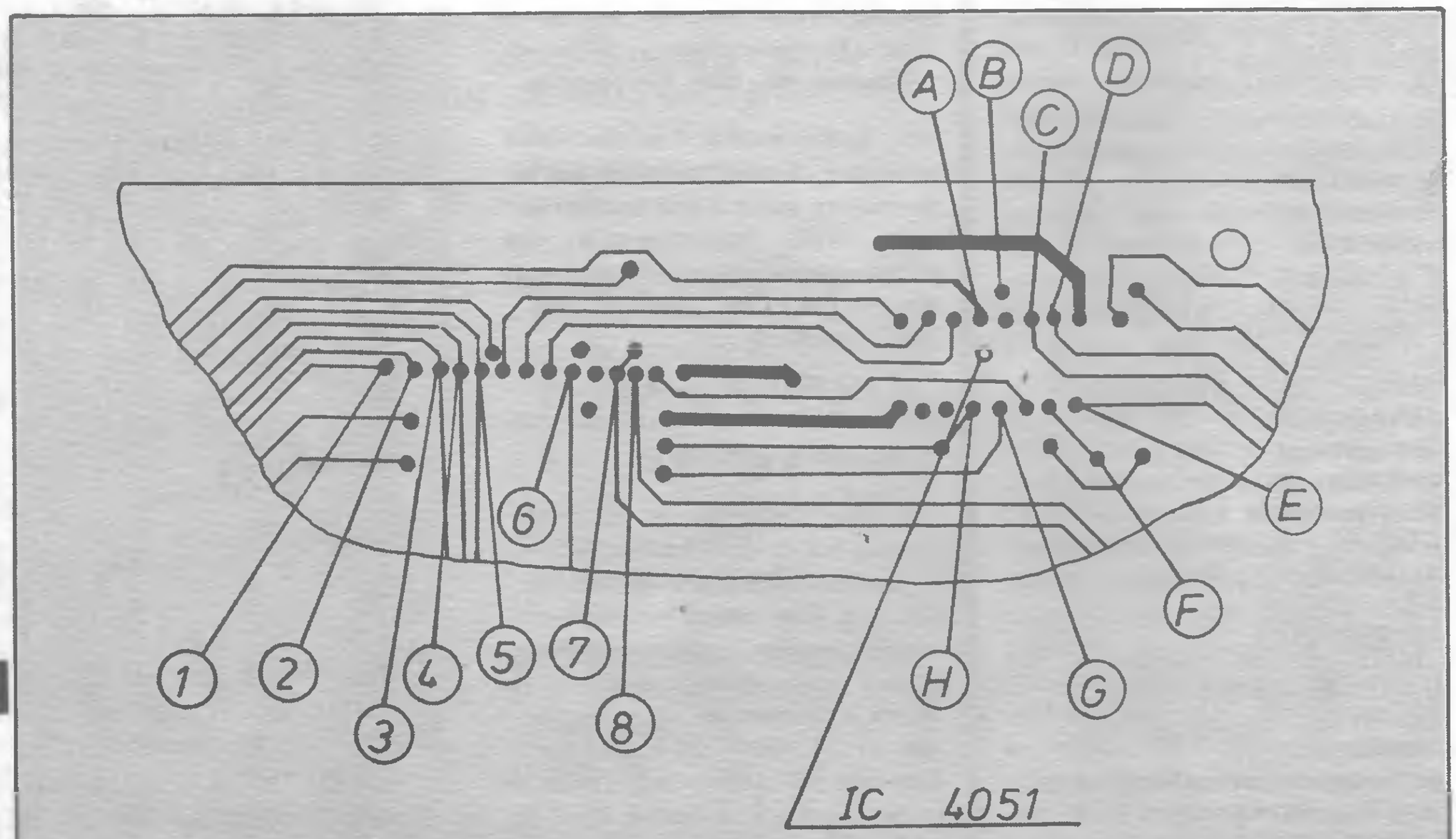
Предлагам един лесен начин за свързване на джойстик към домашния Правец.

Към платката на клавиатурата в посочените точки (фиг. 1) се запоява 16-проводен плосък кабел, не по-дълъг от 60 cm, и се прекарва през един от вентилационните отвори. С подходящ куплунг кабелът се включва към кутия с 10 превключвателя. На фиг. 2 е показано свързването на два от тях. В устройството са използвани 5 ридрелета с работно напрежение 5 V. Превключвателите са осемпозиционни и, понеже трудно се намират, може да ги изработите сами.

За по-удобна работа оформете лицевия панел, както е показано на фиг. 3. С помощта на таблицата (фиг. 4) превключвателите се установяват така, че с джойстика да се подават 5 от кодовете на клавиатурата. На фиг. 3 са избрани следните клавиши: наляво — J, надясно — K, нагоре — I, надолу — M, и бутон на джойстика — B.

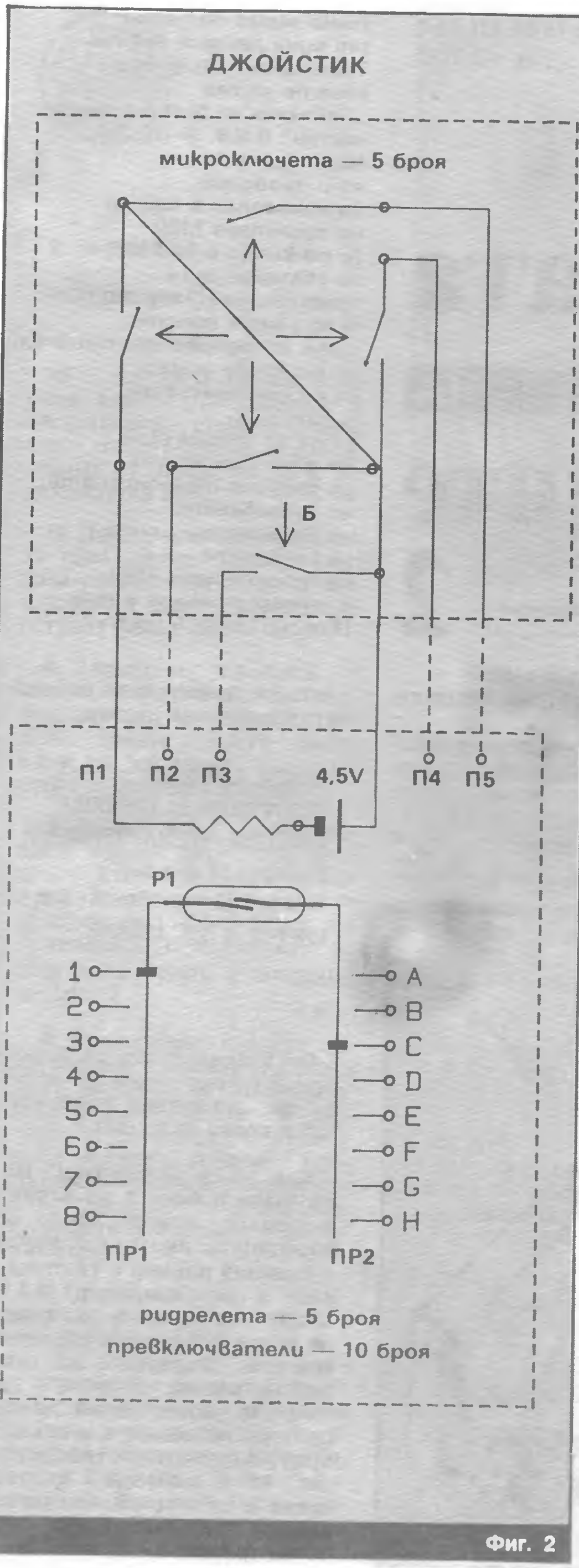
По този начин с джойстика могат да се управляват всички програми, които не изискват повече от 5 клавиша. С добавяне на още превключватели може да се включи джойстик за втори играч или да се увеличи броят на бутоните.

Описаното устройство използвам повече от две години. То не влошава работата на компютъра и ми е спестило ремонта на клавиатурата (когато децата играят с компютъра).

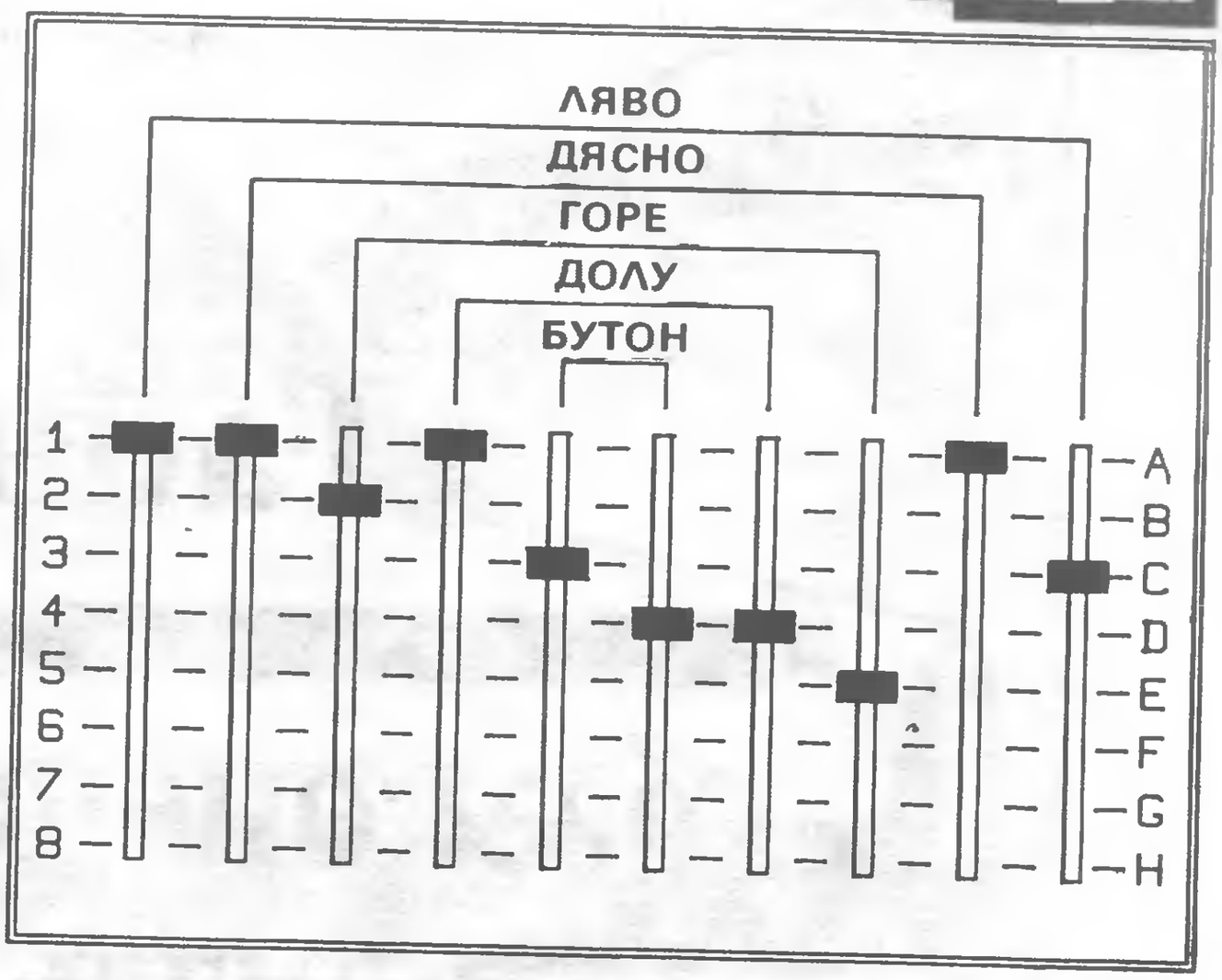


Фиг. 1

ДЖОЙСТИК



Фиг. 2



Фиг. 3

	A	B	C	D	E	F	G	H
1	K	7	J	M	U	8	Y	SPA
2	9	N	T	6	I	L	H	<
3	[	5	R	B	O	0	G	>
4	*	V	F	4	P	?	E	↑
5	]	3	D	C	Ю	-	W	→
6	+	X	Q	2	Є	Z	S	↓
7		1	ОСВ	Z	DEL	↵	A	←
8	F1			МК		⇧R		⇧L

Фиг. 4

Ст.н.с. к.т.н. АЛЕКСАНДЪР ДАСКАЛОВ

Тайната на безконечните ленти — 2

Няма нищо по-лесно от една решена задача. Тази мисъл ми дойде, когато четях „Тайната на безконечните ленти" в КВ. 9—10. 89. Истината е, че с проблема за износените ленти на принтера М80 (а по-късно и на М88) се сблъсках още преди няколко години. И го реших просто, така че тогава не помислих, че от това могат да се интересуват и други, за да го публикувам. Но сега съм готов да споделя технологията на залепването на безконечни ленти за касетите на принтерите М80 и съвместимите с тях (Епсон, Стар и др.).

КВ–52

1. Необходими инструменти

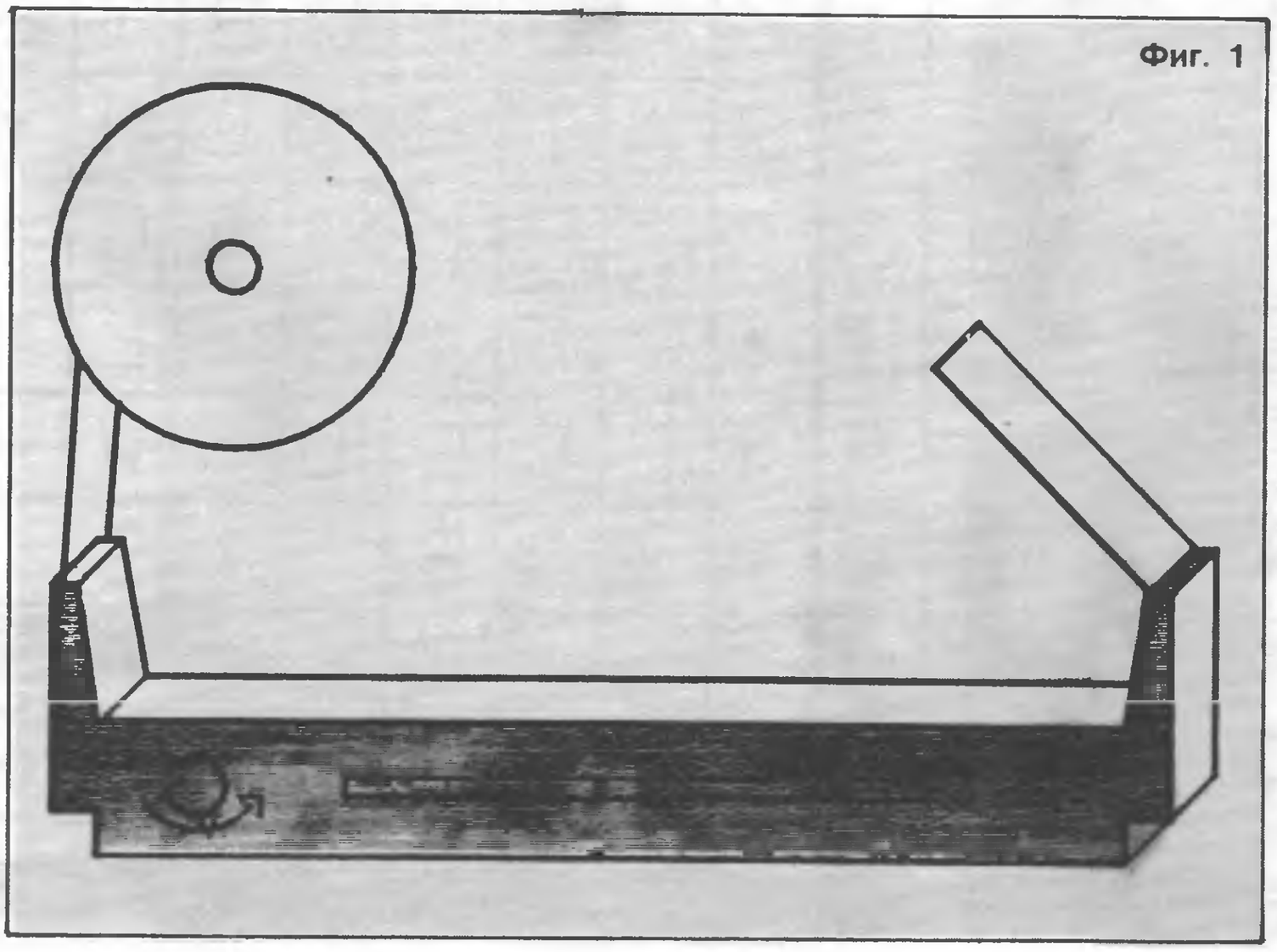
Те са:

- Малко лабораторно менгеме.
- Обикновен електрически поялник.
- Стандартна 13-милиметрова лента за пишеща машина.

Ако в лентите има тайна, според мен тя е в материала на лентата. Затова за нея ще отделя малко повече внимание. Машинописните ленти се произвеждат от завод „Хемус" – Бургас, по БДС 15585–82. Всяка машинописна лента се придружава от етикет със следния надпис:

ЛЕНТИ БАГРЕЩИ
ТЕКСТИЛНИ ЗА ПИШЕЩИ
МАШИНИ (ЧЕРНА, ЧЕРВЕНА)
Трайност 24 месеца
13 мм ± 0.5 mm 10 m ± 0,2 m
13-3-1-2-3-10 БДС 15585–82

Фиг. 1

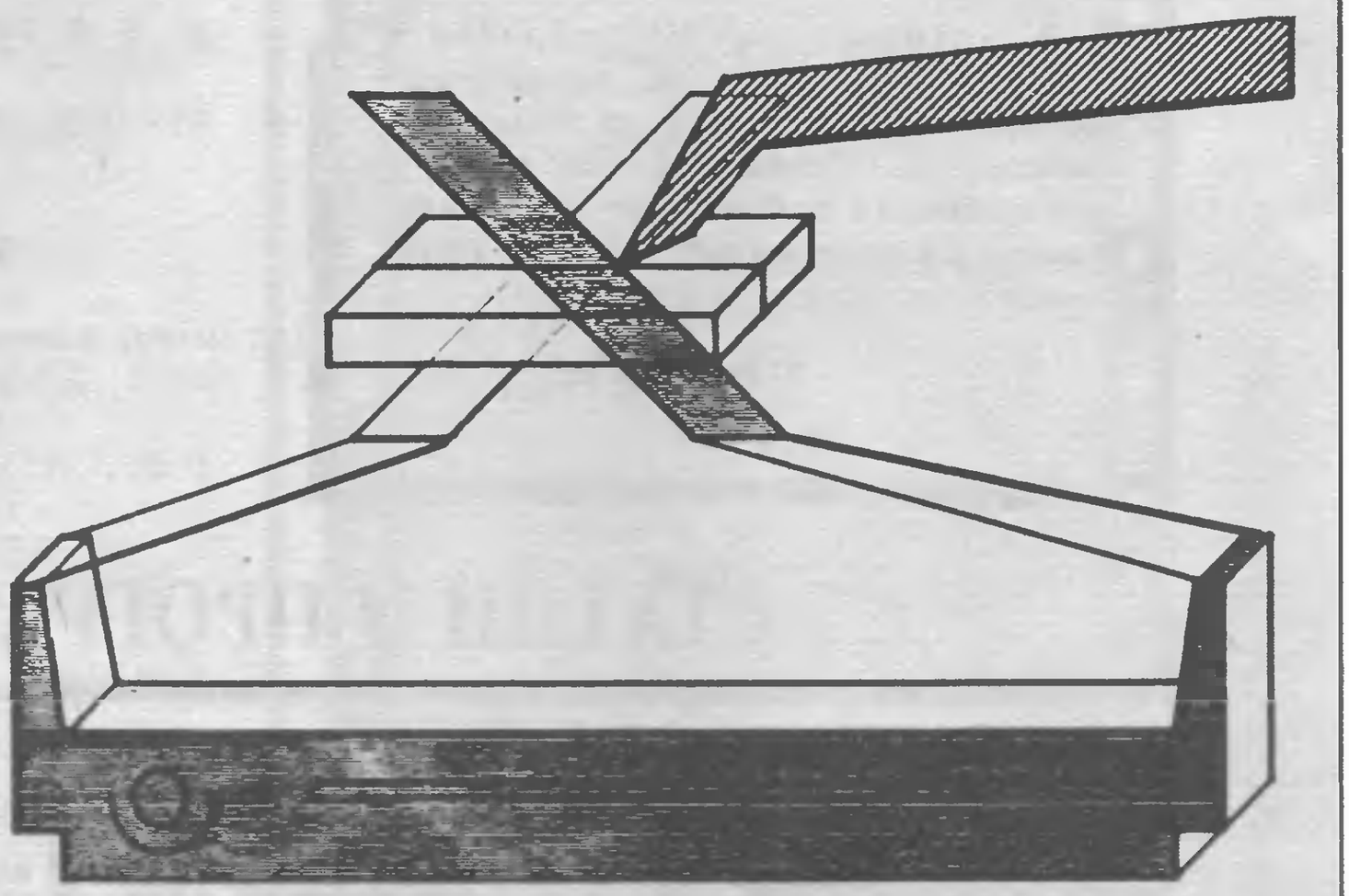


* * *

От гледна точка на безкрайност тези ленти се делят на две – на такива, които стават и които не стават.

Как да ги разпознаем. Изследването показа, че всичко зависи от второто число от поредицата: ако то е 3, както е в нашия пример – тя става. Ако е 2 – например 13-2-1-2-3-10 – не става, т. е. може да се ползва само за пишеща машина. Първата е на найлонова основа, втората – на памучна, което може да се провери веднага с клечка кибрит. Ако лентата тлее и пуши, тя е памучна. Ако се свива и се сгърчва, без да се запали – тя е точно тази, която ни трябва.

Фиг. 2



2. Залепване

Алгоритъмът е:

● Запрятаме ръкави.

● С малка отвертка отваряме капака на касетата и отстраняваме старата лента, като внимателно я освобождаваме от притискащия и задвижващия валц.

● Прекарваме новата лента през десния процеп, след това през левия, така че свободният ѝ край и ролката да останат навън (фиг. 1).

● Затваряме касетата с капака и започваме да въртим валчето на водещото колело, докато цялата лента влезе в магазина и навън останат само двата края.

● Двата края поставяме един върху друг под прав ъгъл. (Ако искаме да се получи Мьобиусова лента, ги пресукваме.) След това ги притискаме с челюстите на менгемето, както е показано на фиг. 2

● Прекарваме човката на нагорещен поялник няколко пъти по ръба на менгемето, докато двете ленти са разтопят и се залепят, а излишните парчета паднат. Продължаваме да нагряваме, докато шевът почти се изравни с горните повърхности на челюстите.

● Освобождаваме челюстите и изправяме лентата – тя е готова за работа.

Ако искаме да не си цапаме ръцете, можем да ползваме леките полиетиленови ръкавици за еднократна употреба, които фирмата „Pelikan“ доставя с всяка нова лента – дано завод „Хемус“ разбере намека ми.

Бележки на другите участници в конкурса и на редакцията

1. Приблизително същото решение изпрати Росен Рашев, студент по информатика II курс в Софийския университет. Той е забелязал важна подробност – производителите на ленти ги лепят **под ъгъл 30 градуса.** И наистина – по този начин шевът става по-дълъг, а в замяна на това във всеки момент върху него попада само една игла от печатащата глава. Той уточнява, че по-прецизно се работи с 20-ватов маломощен поялник.

2. Инж. Николай Дамов от Петрич прилага същата технология, като ползва индукционен поялник. Той съобщава друга важна информация – че е разработил технология как **да напоява отново с мастило изсъхналите, но здрави ленти.** За тези, които се интересуват, той праща пълния си адрес:

2850 Петрич
ул. Първи май 1
дом. тел. 47-11
инж. Николай Дамов

Редакцията също се интересува от тази технология и ако инж. Дамов ни я предостави (срещу приличен хонорар, разбира се), ние ще я публикуваме.

3. Редакцията кани Росен Рашев и инж. Дамов да получат обещаното възнаграждение.

■

В отговор на многобройните читателски писма и нарасналия интерес към програмите за домашния компютър Правец-8Д за нашите нови абонати публикуваме подробен списък на материалите на тази тема, поместени досега в списанието.

КОМПЮТЪР ЗА ВАС

СТАТИИ И ПРОГРАМИ ЗА

ПРАВЕЦ–8Д

ПУБЛИКУВАНИ В СП. „КОМПЮТЪР ЗА ВАС“

I. ТЕОРИЯ НА ПРОГРАМИРАНЕТО

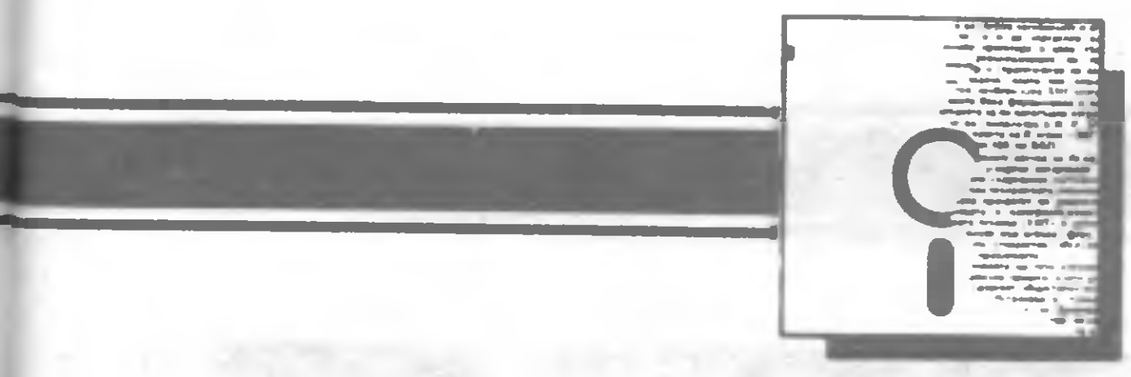
1. Домашният компютър Правец-8Д. 1986, № 2 – 3, 23 – 25.
2. Правец-8Д. 1986, № 4, 26 – 28.
3. Адреси в нулевата страница. 1987, № 8 – 12, 1988, № 1 – 8.
4. Управляване на звуковия процесор. 1988, № 7 – 8, с. 46.
5. Адресиране на екрана. 1988, № 7 – 8, 46 – 47.
6. Команди за ДОС-8Д. 1989, № 5 – 6, 50 – 51.
7. Мнимият RESET. 1989, № 5 – 6, с. 53.
8. Скрол в обратна посока. 1989, № 5 – 6, с. 53.
9. Цифрови входове и изходи за Правец-8Д. 1989, N 7 – 8, с. 27.
10. Как Правец-8Д възприема командата ONERR GOTO. 1989, № 7 – 8, с. 28.
11. От ДОС 3.3 към ДОС-8Д. 1989, № 9 – 10, 37 – 39.
12. Функцията & (амперсанд) при Правец-8Д. 1989, № 9 – 10, с. 40.
13. Програмируемо реле. 1989, № 9 – 10, с. 41.

II. ХАРДУЕР

1. Контролер за паралелен обмен между Правец-82 и Правец-8Д. 1987, № 6, 27 – 31.
2. Присаждане на схема. 1988, № 4 – 5, с. 53.
3. Свързване с външна платка. 1988, № 4 – 5, 53 – 54.
4. София-85 като монитор. 1988, № 7 – 8, 44 – 45.
5. Контролер за флопидисково устройство към Правец-8Д. 1989, № 1 – 2, 41 – 49. Допълнение: 1989, № 5 – 6, с. 7.
6. Под капака на Правец-8Д. 1989, № 5 – 6, 46 – 49.
7. Кабел за връзка между Правец-8Д и едноцветен монитор. 1989, № 7 – 8, с. 44.

III. СЛУЖЕБНИ И ПОМОЩНИ ПРОГРАМИ

1. Копиране на програми. 1986. № 11, с. 28.
2. Командата CATALOG за Правец-8Д. 1986, № 12, 25 – 26.
3. Универсален инструмент. 1987, № 2, 29 – 31.
4. Към универсалния инструмент. 1987, № 10 – 11, с. 59.
5. Редактор за знаците на Правец-8Д. 1987, № 3, 23 – 24.
6. Търсач. 1988, № 3, 25 – 28.
7. Монитор. 1988, № 9 – 10, 60 – 62.
8. Знаков редактор. 1988, № 11 – 12, 51 – 52.
9. Програми на Бейсик от Правец-82 на Правец-8Д. 1989, №5 – 6, с. 52.



10. Функционални клавиши. 1989, № 7 – 8, 50 – 51.
11. Графика на принтер М80. 1989, № 9 – 10, с. 44.

IV. ПРИЛОЖНИ ПРОГРАМИ

1. Подготовка на електронни схеми и други графични документи на Правец-8Д. 1986, № 6, 23-25.
Допълнения: 1986, № 11, с. 32 и 1987, № 9, с. 22.
2. Видеоанимация. 1986, № 7, 24 – 26.
Допълнения: 1986, № 11, с. 32 и 1987, № 9, с. 22.
3. Програма за случайни числа. 1986, № 8, с. 29.
4. Копиране на програми. 1986, № 11, с. 28.
5. Музикална клавиатура за Правец-8Д. 1987, № 6, 25 – 26.
6. Задача за търговския пътник. 1987, № 10 – 11, 33 – 36.
7. Микроплан. 1987, № 10 – 11, 43 – 45.
8. Биоритми. 1988, № 4 – 5, с. 45.
9. Шрифт за Правец-8Д. 1988, № 4 – 5, с. 46.
10. Математика за 8Д. 1988, № 7 – 8, 41-42.
11. Принтер за Правец-8Д. 1989, № 1 – 2, с. 50.
12. Електронен календар-бележник. 1989, № 3 – 4, 57 – 60.
13. Редактор HIRES. 1989, № 5 – 6, 60 – 62.
14. Пиано с памет. 1989, № 5 – 6, с. 63.
15. Електронен дует. 1989, № 7 – 8, 52 – 53.

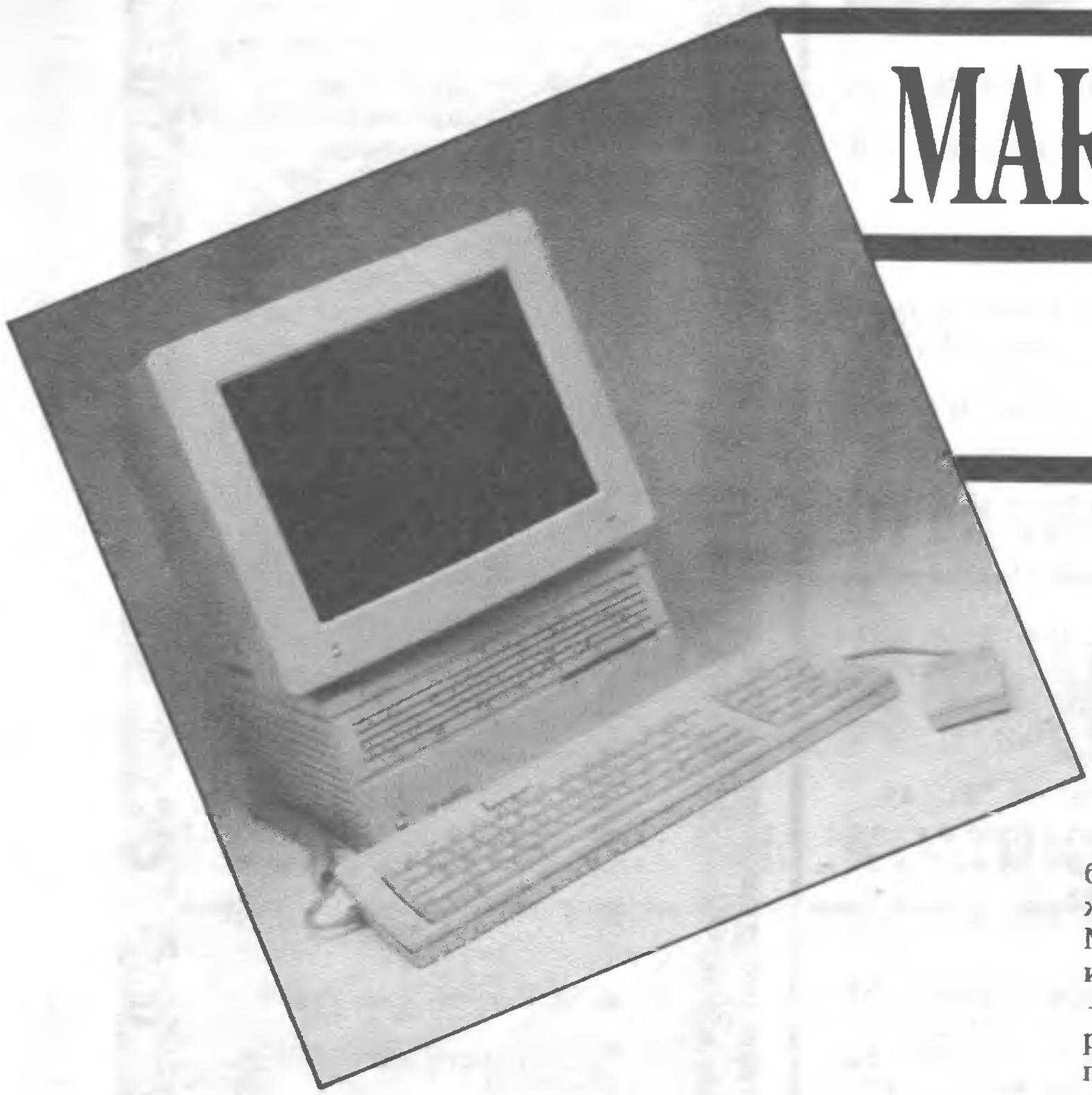
V. ИГРИ

1. Танкова битка. 1986, № 9 – 10, 55 – 57.
Допълнения: 1986, № 11, с. 32 и 1987, № 9, с. 22.
2. Лабиринт. 1987, № 9, 22 – 24.
Допълнение: 1987, № 12, с. 18.
3. Коте на кръстопът. 1988, № 1 – 2, 47 – 50.
4. Препятствие. 1988, № 4 – 5, 47 – 49.
5. Так-тикс. 1988, № 4 – 5, 49 – 50.
6. Слалом – 8Д. 1988, № 6, с. 30.
7. Лакомата змия. 1988, № 7 – 8, 39 – 40.
8. Стена. 1988, № 7 – 8, 43 – 44.
9. Мускетари вкъщи. 1988, № 11 – 12, 52-55.
10. Обърни се, човече! 1989, № 1 – 2, 50 – 52.
11. Слалом. 1989, № 3 – 4, 60 – 61.
12. Случка в градината. 1989, № 5 – 6, 56 – 57.
13. Приказка за 8Д. 1989, № 5 – 6, 58 – 59.
14. Тетрис. 1989, № 7 – 8, 54 – 56.
15. Мери. 1989, № 7 – 8, с. 57.
16. Няма време! 1989, № 7 – 8, с. 58.
17. Червейче. 1989, № 7 – 8, с. 59.

i – борса • i – борса • i – борса •

МАКИНТОШ IIci

ДИМИТЪР ВАВОВ

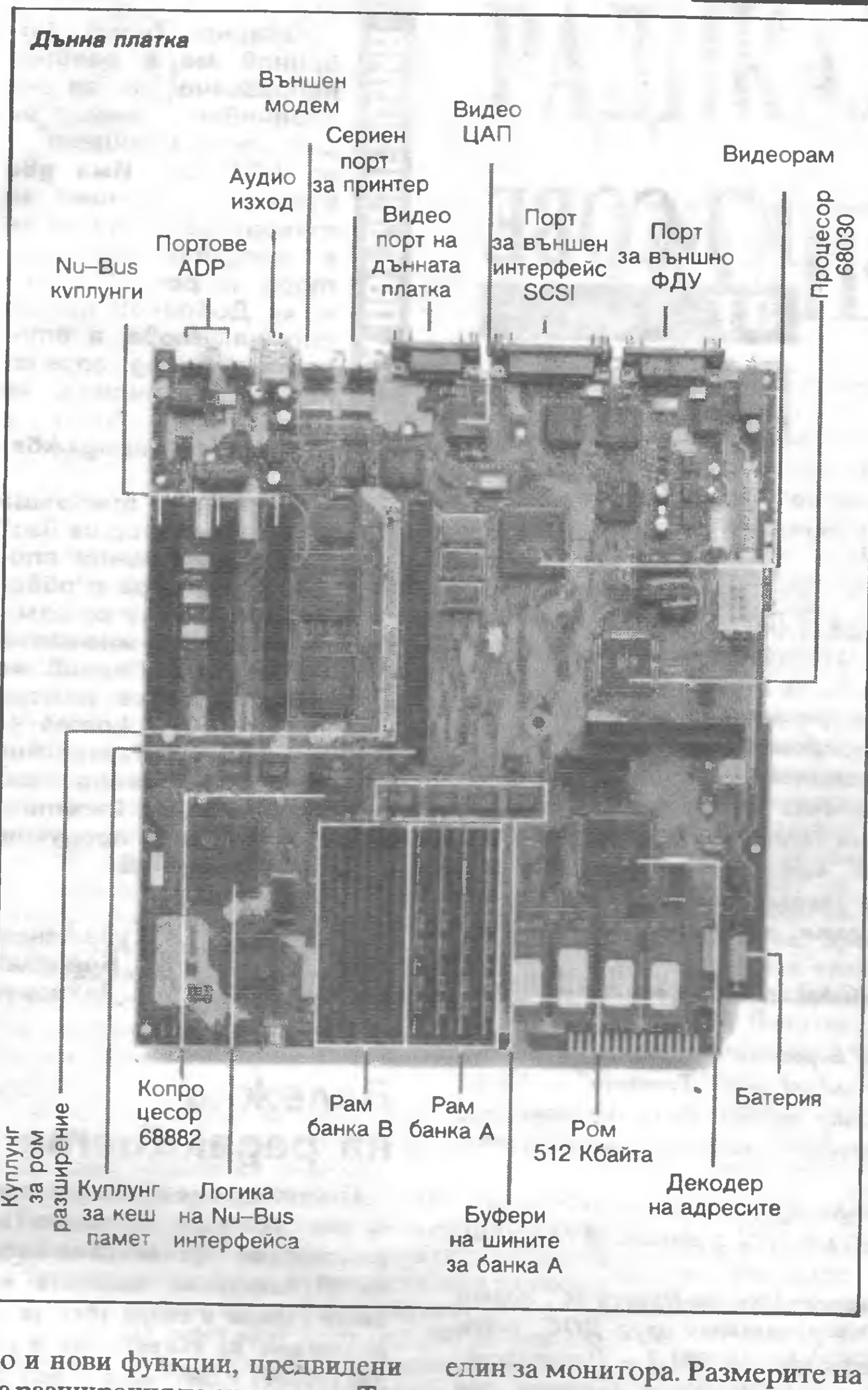


Макинтош IIci е символ на авангарден подход и технологични постижения в микроелектрониката. Той е пряк наследник на обявения само преди шест месеца най-силен модел на фамилията – Мак IIcx, но предлага 35-90% по-висока производителност от него! Работи с най-мощния засега 32-битов микропроцесор – Моторола 68030, и съответния аритметичен копроцесор 68882. За разлика от моделите Мак IIx и IIcx честотата тук вече е 25 MHz вместо 16 MHz.

Мак IIci продължава концепцията на „Епъл“ за висока степен на интеграция и включва пет нови (от общо 16) специализирани схеми ASIC (Application Specific Integrated Circuit) от типа VLSI. Освен процесора и копроцесора една схема реализира интегрирано видео Nu-Bus (директно свързване на 12" монохромен – 640x870 и 13" цветен монотор – 640x480 точки), втора – интерфейса на Епъл SCSI (Small Computer System Interface) за управление на твърд диск, трета – флопидиск контролер, четвърта – прочутия музикален чип на Епъл, с който Макинтош свири като оркестър (ASSC – Apple Stereo Sound Chip), пета – сериен комуникационен контролер, шеста – Nu-Bus комуникационен интерфейс, седма – декодер на адресите в паметта, и т.н. Само няколко думи за стандарта Nu-Bus, исторически наложил се при минимашините. Разработка на Мазачузетския технологичен институт, собственост на „Тексас инструмънтс“, характеризиращ се с независимост от микропроцесора. Позволява обмен на 8-, 16- и 32-битови данни, 32-битово адресиране, блоков режим за обмен при скорост 10-36 Мбита/сек. На дънната платка има изведени пет съединителя – три тип Nu-Bus и по един за разширение на ром и кеш-рам памети.

Оперативната памет е от 4 до 8 Мбайта, монтирана в банки по 1 Мбайт на осем цокъла. Повишено е бързодействието и на паметта – времето за достъп до нея е 80 ns. Използван е метод на странициране – режим „burst“ на процесора, който ускорява допълнително работата, като осъществява обмена на данни с по-малко цикли. Така например за прехвърлянето на пакета от 32-битови данни са необходими само 11 цикъла вместо 20 при Мак II.

Ром паметта в сравнение с досегашните модели нараства от 256 на 512 Кбайта, което е показателно за развития системен софтуер. В рамките на четири схеми по 128 Кбайта са включени и набор помощни програмни средства Toolbox, графичните подпрограми QuickDraw 32, как-



то и нови функции, предвидени за разширения на системата. Това позволява адресиране на памет до 4 Гбайта и поддържане на 786 432 цвята от палитра с 16,8 млн. цвята...

Стандартно конфигурацията включва едно флопидисково устройство 1,4 Мбайта и твърд диск 80 Мбайта — и двете в наложения преди години от „Епъл" формат 3,5". Задният панел е наситен с изводи — за два RS422, два порта ADB, един SCSI, един за външно флопидисково устройство, един за аудио стерео, един за монитора. Размерите на компютъра са 302x365x140 mm, масата му — 6,2 kg.

Използването на кеш-паметта ускорява работата на системата с 20-30%, но струва пари.

За работа с монитори над монохромното 640x870 и за избор над 256 цвята е необходима допълнителна платка — например за 21" монитори с разделителна способност 1152x870. Това вече струва много пари при Макинтош.

ВОТ НА ДОВЕРИЕ

На общо събрание на редакциите на сп. "Компютър за вас" и на сп."Направи сам", проведено на 17 януари 1990 г., единодушно бе гласуван вот на доверие на главния редактор инж. Георги Балански.

ПРАВЕЦ-8С

Ето и допълнителната информация, която получихме от Комбината по микропроцесорна техника за домашния компютър Правец-8С.

Конфигурацията, която се продава за 1230 лв., включва:

– оперативна памет 128 Кбайта и 16 Кбайта постоянна памет;
– вграден флопидисков контролер за включване на две дискетни устройства;
– вграден паралелен интерфейс тип Centronics за включване на принтер;
– вграден модул RGB за включване на компютъра към цветен монитор или телевизор;
– куплунг за джойстик;
– клавиатура;
– едно флопидисково устройство – едностранно с 140 Кбайта капацитет;
– документация: паспорт, техническо описание, ръководство за работа с Бейсик (книга), ДОС 3.3 (книга) с дискета.

На дънната платка е вграден и сериен интерфейс RS 232, но в стандартната конфигурация липсват следните елементи:

– ИС 8250 (СМ 611);
– ИС 75154;
– ИС 75150;
– ИС 74LS90;
– кварцов генератор 18,432 MHz;
– пет резистора.

Принципната схема на серийния интерфейс е показана на лист 4 (долната половина на схемата) в приложение 4 на техническото описание.

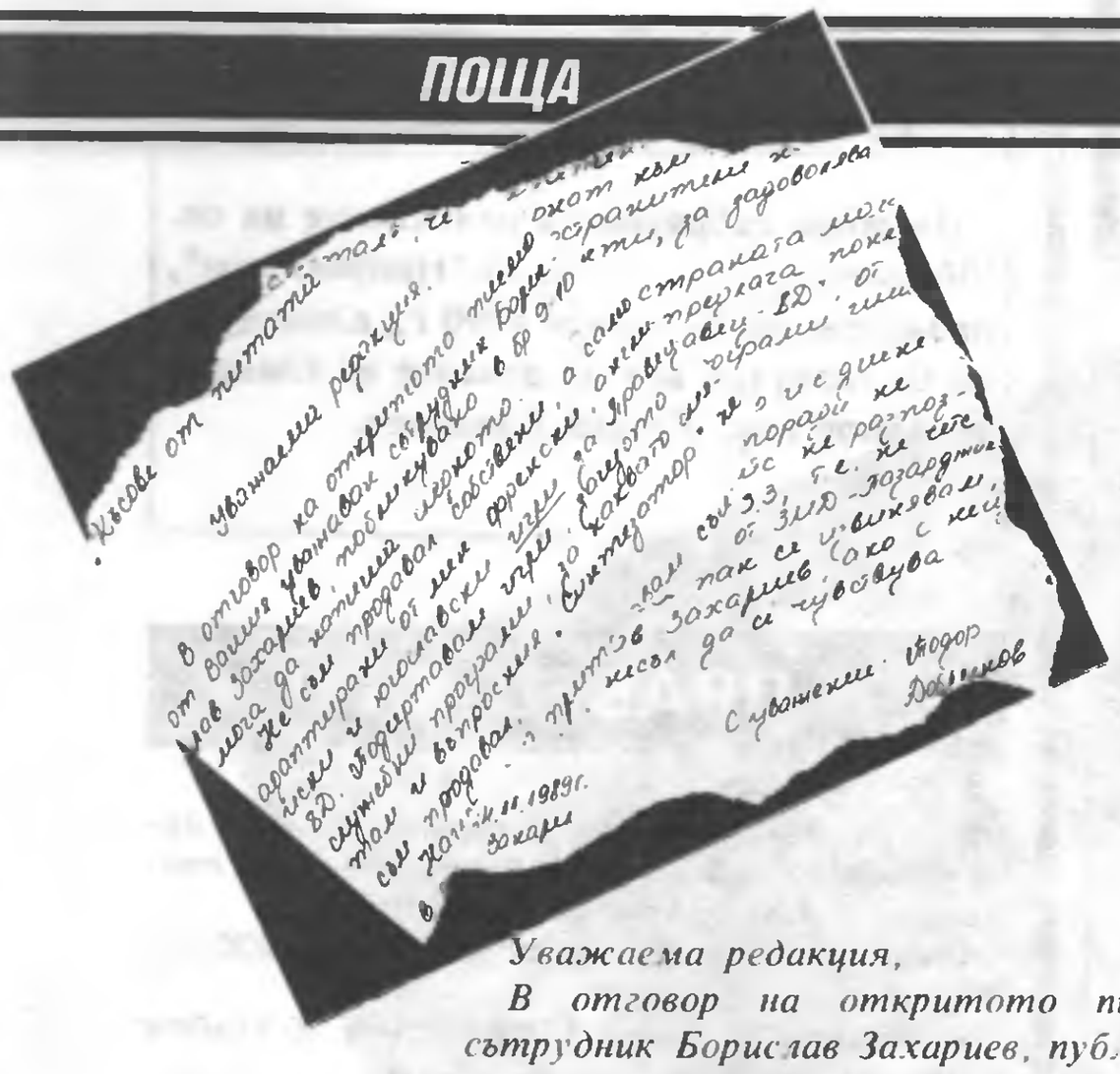
Тези елементи ще се монтират срещу допълнително заплащане във всички поделения на ИП „Системинженеринг". Като допълнителна услуга ще се монтира и куплунг за включване на втори джойстик.

Понеже ИП „Системинженеринг" има поделения в почти всички големи градове, даваме адресите само в София и Правец.

София
бул. „Г.Димитров" 145
тел. 31-51-64
ул. „Цар Симеон" 61
тел. 83-37-15

Правец
тел. 23-51, вътр. 317.

КВ–57



КЪСОВЕ ОТ ПИТАТА

Уважаема редакция,

В отговор на откритото писмо от Вашия уважаван сътрудник Борислав Захариев, публикувано в бр. 9—10, мога да напиша следното.

*Не съм продавал собствени, а само адаптирани от мен френски, английски и югославски **игри** за Правец-8Д. Подчертавам игри, защото служебни програми, за каквато смятам и въпросния „Синтезатор", не съм продавал.*

Наистина притежавам същия в оригиналния му вариант и го доработвам за ползване като телефонен секретар.

*Но тъй като ви пиша не за оправдаване, а с цел изясняване на нещата, ще спомена, че и аз срещнах програми, закупени от „Тошо". Разбира се, осакатени или неработещи, които оправях. Въпреки че нямам брат близнак, нито пък способност да се превъплъщавам в друг образ. За справка прилагам визитната картичка на клиент, пред когото трябваше да доказвам с паспорта си, че **аз** съм Тошо.*

Колкото до контролера и ДОС-8Д, също нямам нищо общо. Дори не ги ползвам.

А може би това да е бил друг Борислав Захариев... Знам ли?

Впрочем, ако искаме да не се въдят още „Тошовци" и други, смятам, че компетентните органи трябва да се обърнат към производителите и разпространителите на програмни продукти, за задоволяване на пазара.

Не зная някъде в страната магазини, в които да се предлага поне една програма за Правец-8Д, от 1982 г. досега. А програми има много!

Подобно е положението и с дискетите за Правец-8С, който поради немонтиран интерфейс не разпознава друг ДОС, освен 3.3(1), т. е. не чете игрите, предлагани от ЗМД — Пазарджик.

И накрая все пак се извинявам пред Борислав Захариев, ако с нещо съм допринесъл да се чувствува обиден.

С уважение:
Тодор Добринов

14. 11. 1989 г. Вакарел

Б. Р. ... Авторът греши, защото зареждането на ПроДос няма нищо общо с „интерфейси".

МНЕНИЕ НА ПОТЪРПЕВШИЯ

Навярно Тодор Добринов ме е разбрал неправилно — аз не обвинявам него за разпространението на ДОС—8Д. Има две възможни причини за твърдението му, че не е продавал синтезатора на реч. Първата е, че Добринов не „си спомня" това, а втората (според гореизложеното писмо), че някой друг „Тошо" е извършвал продажбата.

За да не се превръща сп. „Компютър за вас" в арена за лични спорове, ще спра с обвиненията и ще се солидаризирам с мнението на Тодор Добринов, че така и не се намери организация, която да поеме търговския риск по разпространяването на касети с програмни продукти за Правец—8Д.

30. XI. 89

С уважение:
Борислав Захариев

●

Бележка на редакцията

Отново искаме да подчертаем, че сме далеч от желанието да разследваме търговията по битака. Публикуваме мненията на двете страни в спора (без да ги подлагаме на съкращения и редактиране) единствено с цел да привлечем вниманието върху проблемите при разпространяването на програмни продукти в условията на необясним дефицит.

КВ–58

ИЗВИНЕНИЕ

В КВ 01.89 нашият редовен сътрудник Веселин Бончев публикува под заглавието „По-доброто решение" листинга на къса остроумна програма, разработена от Валери Трифонов. Името на автора бе изрично цитирано, но се оказа, че програмата е предложена без съгласието на автора — нито за публикуването ѝ, нито за промените, направени в нея. Редакцията се извинява на Валери Трифонов за този пропуск. Ако някой от читателите прояви интерес към оригиналния листинг, нека се обърне към редакцията.

КВ

РАДИКАЛНА ПОЛИМЕРИЗАЦИЯ

ИВАН МИТЕВ,
ДИМО ДИМОВ,
АНГЕЛ МИТЕВ

Програмата е разработена за Правец-82 и съвместими с него компютри. Предназначена е за използване в урочната работа по химия за X клас във всички средни училища. Някои части на програмата са разработени по-задълбочено, за да могат да се използват в обучението по специална химическа технология в техникумите, СПТУ и УПК по полимерпреработваща промишленост.

В редовете от 2 до 45 са разположени подпрограмите. Менюто и управляващият блок със 7 независими файла са разположени в редове от 50 до 80. Седемте файла от менюто се реализират в съответната си стотица от програмните редове, съответно от 100 до 700. Програмата заема 23 сектора на дискета.

Формулите на химическите съединения са представени структурно. Химическите реакции са показани в динамика чрез движение на атоми и електрони, възникване и изтриване на валентни връзки.

След стартиране на програмата на екрана се показва менюто с възможност за демонстрация на седем химически реакции:

1. Бензоилперокис – от ред 100 до ред 180. Показва се формулата на бензоил перокиса, разпадането му на два бензоатни радикала и тяхната дисоциация на два фенилови радикала и две молекули въглероден двуокис.

2. Калиев персулфат – от ред 200 до ред 270. Показва се формулата на калиевия персулфат и две молекули вода. Осъществява се миграция на два протона от водата към прекисните кислороди на персулфата с образуване на две молекули кисел калиев сулфат и два хидроксилни радикала.

3. Полимеризация – от ред 300 до ред 330. Показва се свободен радикал с обща формула и молекула винилов мономер. Следва активиране на мономерната молекула и превръщането ѝ в мономерен радикал. Нарастването на веригата е показано чрез последователното присъединяване на две мономерни молекули към мономерния радикал и превръщането им в макрорадикал.

4. Предаване на веригата – от ред 400 до ред 450. Показва се макрорадикал и обща формула на регулатор на молекулната маса. Следва деактивиране на макрорадикала чрез откъсване на протон от регулатора на молекулната маса и превръщането му в свободен радикал. Свободният радикал активира молекула мономер, а образувалият се мономерен радикал присъединява нова молекула мономер. Деактивирането на новия радикал се извършва с нова молекула регулатор на молекулната маса, който се превръща в нов свободен радикал.

5. Разклонения на веригата – от ред 500 до 570. Показва се елементарно звено от полимерна верига на винилов мономер и свободен радикал. Радикалът откъсва протон от макромолекулата, която се превръща в макрорадикал, а радикалът се деактивира. Макрорадикалът присъединява две молекули мономер и се деактивира с предавател на веригата.

6. Рекомбинация – от ред 600 до ред 630. Показват се два макрорадикала, свободните електрони на които се сдвояват с образуване на валентна връзка, в резултат на което се получава една молекула полимер.

7. Диспропорциниране – от ред 700 до ред 750. Показват се два макрорадикала, които се деактивират чрез миграция на протон от единия радикал с образуване на две молекули полимер – едната с наситен край, а другата с ненаситен.

Демонстрацията на всяка химична реакция завършва с изчакване на две последователни натискания на произволен клавиш. След всеки отделен кадър от реакцията се изчаква натискане на произволен клавиш. По време на всяка пауза може да се натисне И или I, като от кадрите се излиза в менюто, а от менюто – в команден режим на Бейсик, без да се изтрива програмата от паметта.

КВ–59

```
1 TEXT:HOME:GOTO 50
2 HPLOT X+22,82 TO X+26,82 TO X+26,86 TO X+22,86 TO X
  + 22,90 TO X+26,90
3 HPLOT X,71 TO X,85:HPLOT X+1,70 TO X+6,70:HPLOT X+1,86
  TO X+6,86:HPLOT X+7,71 TO X+7,73:HPLOT X+7,85 TO X+7,83
4 HPLOT X+11,Y TO X+11,Y+16:HPLOT X+18,Y TO X+
  18,Y+1:HPLOT X+11,Y+8 TO X+18,Y+8:RETURN
5 HPLOT X+3,90 TO X+3,106:HPLOT X,126 TO X,110 TO X+7,110
  TO X+7,118 TO X,118 TO X+7,126:HPLOT X+11,124 TO
  X+13,122 TO X+ 13,130 TO X+11,130 TO X+15,130:RETURN
```

```
  6 HPLOT X,86 TO X,70 TO X+7,70 TO X+7,78 TO X,78 TO
    X+7,86:RETURN
  7 HPLOT X+3,64 TO X+4,64 TO X+4,65 TO X+3,65:RETURN
  8 HOME:VTAB 24:PRINTTAB(21-LEN (A$)/2)A$ TAB(101-LEN
    (B$) / 2)B$
  9 POKE 49168,0:WAIT 49152,128:VTAB 1:GET C$:IF C$="I" OR
    C$="M" THENHOME:GOTO 50  10 B$="":RETURN
 11 HOME:POKE 230,32:CALL 62450:HGR:HCOLOR=3:RETURN
 12 FOR P=0 TO M:NEXT:RETURN
 13 HPLOT X,Y TO X,Y+20:HPLOT X+10,Y TO X+10,Y+20:HPLOT
    X,Y+10 TO X+10,Y+10:RETURN
 14 HPLOT X,Y TO X+1,Y TO X+1,Y+1 TO X,Y+1:GOTO 12
 15 HPLOT X+10,Y+5 TO X+10,Y+15
 16 HPLOT X+10,Y+4 TO X+10,Y+1:HPLOT X+9,Y TO X+1,Y:HPLOT
    X,Y+1 TO X,Y+19:HPLOT X+1,Y+20 TO X+9,Y+20:HPLOT X+10,Y+
    19 TO X+10,Y+16:RETURN
 20 Y=70:GOSUB 2:HPLOT X+22,75 TO X+38,75:HPLOT X+22,78 TO
    X+38,78:X=X+42:GOSUB 3:GOSUB 5:GOSUB 8:FOR T=X-84 TO X
    -42:X=T:HCOLOR=3:GOSUB 7:HCOLOR=0:GOSUB
    7:NEXT:HCOLOR=3
 21 GOSUB 7:FOR M=100 TO 0 STEP -10:HCOLOR=3:HPLOT X+22,75
    TO X+38,75:GOSUB 12:HCOLOR=0:HPLOT X+20,75 TO X+40,75:GOSUB 12:
    NEXT:GOSUB 7:HCOLOR=3:HPLOT X-20,78 TO X-4,78:X=X+42:GOSUB 7:RETURN
 25 GOSUB 11:GOSUB 6:X=272:GOSUB 6:FOR X=55 TO 205 STEP
    50:Y=70:GOSUB 3:NEXT:X=55:GOSUB 2:X=205:GOSUB
    2:X=105:GOSUB 5:GOSUB 7:X=155:GOSUB 5:GOSUB 7
 26 FOR L=0 TO 24:Y=78:HPLOT 27+L,Y:HPLOT 77+L,Y:HPLOT
    177+L,Y:HPLOT 227+L,Y:NEXT:FOR L=0 TO 17 STEP 3:HPLOT 11+L,Y:
    HPLOT 253+L,Y:NEXT:A$="МАКРОРАДИКАЛИ":GOSUB 8:RETURN
 27 REM M.МИТЕВ'87
 30 GOSUB 11:Y=40:X=90:GOSUB 16:X=105:GOSUB 13:HPLOT 120,55
    TO 125,55 TO 125,60 TO 120,60 TO 120,65 TO 125,65:HPLOT 120,45
    TO 150,45:HPLOT 120,50 TO 150,50:X=155:GOSUB
    16:X=170:GOTO 13  31 HPLOT 160,65 TO 160,90:RETURN
 33 FOR M=150 TO 10 STEP -10:HCOLOR=3:HPLOT 120,45 TO
    150,45:GOSUB 12:HCOLOR=0:HPLOT 120,45 TO 150,45:GOSUB
    12:NEXT:HCOLOR=3
 35 HPLOT 55,50 TO 85,50:HPLOT 185,50 TO 215,50:FOR L=0 TO
    29 STEP 5:HPLOT 25+L,50:HPLOT 220+L,50:NEXT:RETURN
 42 HPLOT 22,Y TO 22,Y+20:HPLOT 31,Y TO 22,Y+10 TO
    31,Y+20:FOR Y=Y TO Y+6:FOR X=35 TO 45 STEP 10:HPLOT X,Y
    TO X+ 6,Y:NEXT X,Y
 43 HCOLOR=0:HPLOT 36,Y-4 TO 40,Y-4:HPLOT 46,Y-4 TO
    50,Y-4:HPLOT 38,Y-6 TO 38,Y-2:HCOLOR=3:Y=Y-7:X=56:GOSUB 15:X=70
 44 GOSUB 15:HPLOT X,Y+10 TO X+10,Y+10:HCOLOR=0:HPLOT
    X+10,Y+10 TO X+10,Y+5:HPLOT X,Y+10 TO
    X,Y+15:HCOLOR=3:X=84
 45 GOSUB 15:HPLOT 99,Y+10 TO 125,Y+10:X=130:GOSUB
    15:X=205:GOSUB 13:X=220:GOSUB 15:X=235:GOSUB 13:RETURN
 50 PRINTTAB(9)"РАДИКАЛНА ПОЛИМЕРИЗАЦИЯ"
    TAB(49)"========================"
    TAB(89)"1. БЕНЗОИЛПЕРОКИС"
    TAB(89)"2. КАЛИЕВ ПЕРСУЛФАТ"
    TAB(89)"3. ПОЛИМЕРИЗАЦИЯ"
    TAB(89)"4. ПРЕДАВАНЕ НА ВЕРИГАТА"
 60 PRINTTAB(49)"5. РАЗКЛОНЕНИЯ НА ВЕРИГАТА"
    TAB(89)"6. РЕКОМБИНАЦИЯ"
    TAB(89)"7. ДИСПРОПОРЦИОНИРАНЕ"
    TAB(89)"8. ПРОМИШЛЕНИ ПОЛИМЕРИ"
    TAB(129)"?СТАРТИРАНЕ (1-8)"
    TAB(92)"ИЗХОД (I/M)"
 70 POKE 49168,0:VTAB 21:HTAB 9:GET A$:ONVAL (A$) GOTO
    100,200,300,400,500,600,700,800:IF A$="I" OR A$="M" THEN
    HOME:END
 80 PRINTCHR$ (7):PRINTCHR$ (7):GOTO 70
100 GOSUB 11:FOR X=32 TO 248 STEP 216:Y=50:HPLOT X,Y TO
    X+20,Y+20 TO X+20,Y+45 TO X,Y+65 TO X-20,Y+45 TO X-
    20,Y+20 TO X,Y
110 FOR E=0 TO 6.28 STEP .4:HPLOT X+15 *COS (E),(Y+33)+17
    *SIN (E):NEXT E,X:FOR X=52 TO 208 STEP 39:Y=70:HPLOT X,Y
    TO X+20,Y:NEXT:FOR X=77 TO 194 STEP 39:Y=60:GOSUB
    16:NEXT
120 HPLOT 126,65 TO 126,75:HPLOT 165,65 TO 165,75:FOR L=36
    TO 56:HPLOT 80,L:HPLOT 84,L:HPLOT 197,L:HPLOT 201,L:NEXT:
    Y=12:X=77:GOSUB 15:X=194:GOSUB 15:A$="БЕНЗОИЛПЕРОКИС":GOSUB 8
130 FOR M=150 TO 0 STEP -10:HCOLOR=3:HPLOT 130,70 TO
    150,70:GOSUB 12:HCOLOR=0:HPLOT 130,70 TO 150,70:GOSUB
    12:NEXT:FOR M=0 TO 90 STEP 10:HCOLOR=0:Y=55:X=120:GOSUB
    14:X=159:GOSUB 14
140 HCOLOR=3:X=120:GOSUB 14:X=159:GOSUB 14:NEXT:A$="БЕНЗОАТНИ
    РАДИКАЛИ":GOSUB 8:FOR M=150 TO 0 STEP -10:HCOLOR=3:HPLOT 53,
    70 TO 72,70:HPLOT 208,70 TO 228,70:GOSUB 12:HCOLOR=0
150 HPLOT 53,70 TO 72,70:HPLOT 208,70 TO 227,70:GOSUB
    12:NEXT:Y=55:X=120:GOSUB 14:X=159:GOSUB 14:HPLOT 90,70
    TO 112,70:HPLOT 168,70 TO 190,70:HCOLOR=3
160 FOR X=91 TO 169 STEP 78:HPLOT X,68 TO X+20,68:HPLOT X,72
    TO X+20,72:NEXT:FOR M=0 TO 90 STEP 10:HCOLOR=0:Y=65:X=55:GOSUB 14:
    X=223:GOSUB 14:HCOLOR=3
170 X=55:GOSUB 14:X=223:GOSUB 14:NEXT:A$="ФЕНИЛОВИ
    РАДИКАЛИ":GOSUB 8:GOSUB 9:A$="":RUN
200 GOSUB 11:Y=45:GOSUB 42:Y=95:GOSUB 42:HPLOT 135,69 TO
    135,91:Y=0:X=70:GOSUB 15:Y=139:GOSUB 15:FOR L=24 TO 41:
    HPLOT 73,L:HPLOT 77,L:HPLOT 73.94+L:HPLOT 77.94+L:NEXT:
    A$="КАЛИЕВ ПЕРСУЛФАТ":GOSUB 8
210 FOR X=145 TO 200 STEP 5:HPLOT X,55:HPLOT
    X,105:M=25:GOSUB 12:NEXT:FOR Y=69 TO 92 STEP 2:HCOLOR=0:
    HPLOT 135,Y:NEXT:GOSUB 9:HCOLOR=3:X=224:Y=40:GOSUB 14:
    Y=90:GOSUB 14:FOR X=205 TO 146 STEP -1
220 HCOLOR=3:Y=45:GOSUB 13:Y=95:GOSUB 13:HCOLOR=0:Y=45:GOSUB
    13:Y=95:GOSUB 13:NEXT:HCOLOR=3:GOSUB 13:Y=45:GOSUB 13:
    HCOLOR=0:HPLOT 135,68 TO 135,90
230 HOME:FOR M=0 TO 100 STEP 10:X=224:HCOLOR=0:Y=40:GOSUB
    14:Y=90:GOSUB 14:HCOLOR=3:Y=40:GOSUB 14:Y=90:GOSUB 14:
    NEXT:A$="ХИДРОКСИЛНИ РАДИКАЛИ":GOSUB 8:GOSUB 9:A$="":RUN
300 GOSUB 11:GOSUB 6:GOSUB 7:A$="АЕТИЛИРАНЕ":X=42:GOSUB
    20:HPLOT 12,78 TO 25,78:A$="НАРАСТВАНЕ":X=126:GOSUB 20:
    X=210:GOSUB 20
310 HPLOT 120,60 TO 114,60 TO 114,135 TO 120,135:HPLOT
    192,60 TO 198,60 TO 198,135 TO 192,135:HPLOT 204,125:
    HPLOT 205,126 TO 205,135:HPLOT 206,125 TO 209,125
320 HPLOT 210,126 TO 210,135 TO
    211,135:A$="МАКРОРАДИКАЛ":GOSUB 8:GOSUB 9:A$="":RUN
400 GOSUB 11:GOSUB 6:FOR X=12 TO 35 STEP 3:HPLOT
    X,78:NEXT:HPLOT 35,78 TO 55,78:X=58:Y=70::GOSUB 2:
    HPLOT 80,78 TO 100,78:X=103:GOSUB 3:GOSUB 7:GOSUB 5:
    X=177:GOSUB 6:GOSUB 4
405 A$="ПРЕДАВАНЕ НА ВЕРИГАТА":GOSUB 8
410 M=90:FOR Y=62 TO 50 STEP -3:HPLOT 107,Y:GOSUB 12:
    NEXT:FOR X=107 TO 192 STEP 3:HPLOT X,50:GOSUB 12:NEXT:
    FOR Y=50 TO 70 STEP 3:HPLOT 192,Y:NEXT:X=177
```

```
420 GOSUB 7:FOR Y=70 TO 43 STEP -1:HCOLOR=3:GOSUB
    4:HCOLOR=0:GOSUB 4:NEXT:FOR X=176 TO 93 STEP -3:
    HCOLOR=3:GOSUB 4:HCOLOR=0:GOSUB 4:NEXT
430 FOR Y=43 TO 57:HCOLOR=3:GOSUB 4:HCOLOR=0:GOSUB
    4:NEXT:HCOLOR=3:Y=70:X=103:GOSUB 2:GOSUB 9:X=219:GOSUB
    20:HPLOT 188,78 TO 198,78:GOSUB 8:GOSUB 9:A$="":RUN
500 GOSUB 11:GOSUB 6:HPLOT X+11,84 TO X+13.82 TO X+13.90 TO
    X+11.90 TO X+15,90:HPLOT 11,78 TO 31,78:Y=28:X=35:GOSUB
    4:HPLOT 39,67 TO 39,47:Y=70:GOSUB 3
510 HPLOT 35,29 TO 35,43:HPLOT 36,28 TO 41,28:HPLOT 36,44
    TO 41,44:HPLOT 42,29 TO 42,31:HPLOT 42,43 TO 42,41:HPLOT
    57, 40 TO 61,40 TO 61,44 TO 57,44 TO 57,48 TO 61,48
520 HPLOT 39,15 TO 39,25:HPLOT 39,90 TO 39,100:X=100:FOR L=0
    TO 15 STEP 3:HPLOT 39,L:HPLOT 39,100+L:NEXT:GOSUB
    6:GOSUB 7:A$="РАЗКЛОНЕНИЕ НА ВЕРИГАТА"
530 GOSUB 8:FOR X=103 TO 53 STEP -5:M=80:GOSUB 12:HPLOT
    X,64:NEXT:X=40:GOSUB 7:FOR X=35 TO 80:HCOLOR=3:GOSUB
    4:HCOLOR=0:GOSUB 4:HPLOT X+18,64:NEXT
540 X=100:GOSUB 7:HCOLOR=3:GOSUB 4:GOSUB 9:HCOLOR=0:GOSUB 6:
    GOSUB 4:HCOLOR=3:X=82:GOSUB 20:HPLOT 46,78 TO
    65,78:X=166:GOSUB 20:GOSUB 9:X=260:GOSUB 6:GOSUB 4
550 GOSUB 9:FOR L=215 TO 260 STEP 5:HPLOT L,65:GOSUB
    12:NEXT:HCOLOR=0:GOSUB 4:HCOLOR=3:GOSUB 7:FOR X=240 TO
    208 STEP -1:HCOLOR=3:GOSUB 4:HCOLOR=0:GOSUB 4
560 HPLOT X+20,65:NEXT:X=208:GOSUB 7:HPLOT 215,65 TO
    225,65:HCOLOR=3:GOSUB 2:GOSUB 9:GOSUB 9:A$="":RUN
600 GOSUB 25:HOME:VTAB 22:HTAB 14:PRINT "РЕКОМБИНАЦИЯ":FOR
    T=0 TO 24:HCOLOR=3:X=105+T:GOSUB 7:X=155-T:GOSUB
    7:HCOLOR=0:X=105+T:GOSUB 7:X=155-T
610 GOSUB 7:NEXT:HCOLOR=3:HPLOT 127,Y TO
    151,Y:A$="МАКРОМОЛЕКУЛА":GOSUB 8:GOSUB 9:A$="":RUN
700 GOSUB 25:HOME:VTAB 22:HTAB 11:PRINT
    "ДИСПРОПОРЦИОНИРАНЕ":HCOLOR=0:X=155:GOSUB 7:HCOLOR=3
710 FOR D=7.8 TO 4.8 STEP -.2:HPLOT 115+45 *SIN (D),65-30
    *COS (D):NEXT:X=55:GOSUB 7:HCOLOR=0:GOSUB 2:HCOLOR=3:Y=70:GOSUB 3
720 FOR D=4.8 TO 7.8 STEP .2:X=101+44 *SIN (D):Y=55-30
    *COS (D):HCOLOR=3:GOSUB 4:M=100:GOSUB 12:HCOLOR=0:GOSUB
    4:GOSUB 12:HPLOT 115+45 *SIN (D),65-30 *COS (D):NEXT
730 HCOLOR=3:X=155:Y=70:GOSUB 2:FOR T=0 TO
    25:HCOLOR=3:X=55+T:GOSUB 7:X=105-T:GOSUB
    7:HCOLOR=0:X=55+T:GOSUB 7:X=105-T:GOSUB 7:NEXT
740 HCOLOR=3:HPLOT 77,74 TO 101,74:A$="МАКРОМОЛЕКУЛА":GOSUB 8:
    GOSUB 9:A$="":RUN
800 GOSUB 30:HPLOT 185,55 TO 190,55 TO 190,60 TO 185,60 TO
    185,65 TO 190,65:A$="ЕТИЛЕН":GOSUB 8:GOSUB
    33:A$="ПОЛИЕТИЛЕН":B$="'РОПОТЕН'":GOSUB 8
810 GOSUB 30:GOSUB 31:Y=95:X=155:GOSUB 16:HPLOT 170,95 TO
    170,115 TO 178,115:A$="ВИНИЛХЛОРИД":GOSUB 8:GOSUB 33:
    A$="ПОЛИВИНИЛХЛОРИД":B$="'ДЕВИЛИТ'":GOSUB 8
820 GOSUB 30:GOSUB 31:Y=95:X=155:GOSUB 16:HPLOT 170,115 TO
    170,95 TO 180,115 TO 180,95:A$="АКРИЛНИТРИЛ":GOSUB 8:
    GOSUB 33:A$="ПОЛИАКРИЛНИТРИЛ":B$="'БУЛАНА'":GOSUB 8
830 GOSUB 30:GOSUB 31:HPLOT 160,90 TO 180,110 TO 180,134 TO
    160,154 TO 140,134 TO 140,110 TO 160,90:FOR R=0 TO 6.28
    STEP .4:HPLOT 160+15 *COS (R),122+16 *SIN (R):NEXT
840 A$="СТИРОЛ":GOSUB 8:GOSUB 33:A$="ПОЛИСТИРОЛ":B$="'БУСТРЕН'":
    GOSUB 8:GOSUB 9:A$="":RUN
```

МЕНЮ ЗА ДОС–8Д

Тази кратка програма на Бейсик работи под управлението на ДОС—8Д. Тя извежда каталога на дискетата, поставена във флопидисковото устройство, като пред името на всеки файл се извежда буква от латинската азбука в инверсен режим. Ако броят на файловете е по-голям от 24, се чака натискането на клавиш за продължаване на списъка от файлове. Ако клавишът е ОСВ, извеждането на каталога се прекратява. Такова е всъщност действието на командата DIR на ДОС—8Д.

След извеждането на каталога може да се стартира файл само с натискането на латинската буква пред името му. При това редът с името на файла сменя цвета си. Натискането на клавиша RETURN предизвиква ново извеждане на каталога, например от друга дискета. Използването на такова меню е удобно при стартирането на игри или кратки машинни програми, които връщат управлението на главната програма. Естествено, ако се стартира програма на Бейсик, програмата МЕНЮ се заличава.

БОРИСЛАВ ЗАХАРИЕВ

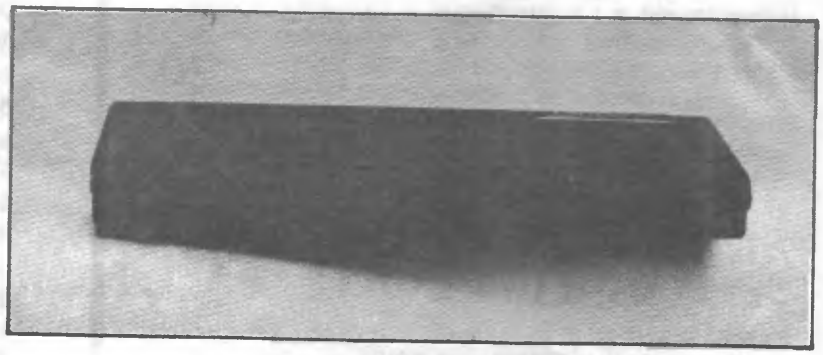
```
10 CLS:POKE $26A,3:INK7:PAPER0:POKE $20C,255
20 LPRINT CHR$(2)"DIR":GOTO 120
30 '
40 REM ИМЕ НА ФАЙЛ ЛИ Е РЕДЪТ?
50 SC=SCRN(5,VT):S=FALSE
60 IF SC>48 AND SC<58 THEN S=TRUE
70 RETURN
80 '
90 REM ПРОЧИТАНЕ НА ИМЕТО ОТ ЕКРАНА
100 N$="":FOR I=9 TO 38
110 N$=N$+CHR$(SCRN(I,VT)):NEXT:RETURN
120 VT=0
130 GOSUB 50:IF NOT S THEN VT=VT+1:GOTO 130
140 GOSUB 50
150 IF S THEN PLOT 0,VT,$A0:PLOT 1,VT,VT+$C1 ELSE 170
160 VT=VT+1:GOTO 140
170 PLOT 2,26,"БУКВА :СТАРТ, RETURN :DIR, DEL :КРАЙ!"
180 PLOT 0,26,22:PLOT 1,26,0
190 PRINT@0,25;:GET C$
200 IF C$=CHR$(127) THEN END
210 IF C$=CHR$(13)  THEN RUN
220 N=ASC(C$):IF N<65 OR N>90 THEN170
230 VT=N-65:GOSUB 50:IF S THEN GOSUB 100 ELSE 170
240 PLOT 8,VT,20
250 LPRINT CHR$(2)"-"N$
260 GOTO 170
```

Съчинение по картинка (от рекламната листовка)

от СЛАВЧО ИВАНОВ



Джобният компютър POQET PC



Концепция

Микрокомпютърът е изграден около централен процесор 80C88, работещ с тактова честота 7 MHz – около 40 процента по-бързо от процесора 8088 на компютрите IBM PC. Системната програма BIOS и софтуерът, обслужващ процесора, са разработени от екип на фирмата Poqet Computer Corporation. Паметта на компютъра се състои от 640 Кбайта ром и 512 Кбайта рам. Едно от главните предимства на джобния компютър са програмите, записани в рома:

- **операционната система MS-DOS 3.30;**
- **GWBASIC,** калкулатор;
- **бележник-календар със звукова сигнализация;**
- **прост текстов редактор;**
- **файлова сервизна програма;**
- **телефонен указател и комуникационна програма.**

Всяка програма се активира със скорост, ограничена от скоростта на достъпа на данните в рома и на трансфера на данни по шините – без загубата от време за достъп до твърд диск или дискета. Тези програми могат да се активират от прозоречно меню (pop-up menu), което се извиква със специален „горещ" клавиш – гоест активен по всяко време.

В кутията на Poqet PC са оставени два процепа, които съответстват по функция на две дискетни устройства. В тях се поставят карти с външна памет с размери приблизително на визитна картичка и с обем от 32 до 512 Кбайта. Те могат да бъдат два вида: ром с предварително записани приложни програми и рам за съхраняване на данни. Главното предимство на тези носители е, че програмите изпълняват директно от тях, без да се налага предварително да бъдат въвеждани в паметта. По този начин се печели бързодействие и голяма част от оперативната памет с обем 512 Кбайта остава на разположение на външната програма. Истина е, че посредством външно дискетно устройство програмите могат да се изпълняват и по конвенционалния начин, но тогава те трябва да се въведат в паметта, като се загуби едно от съществените предимства на концепцията на Poqet PC – по-голямата част от оперативната памет от 512 Кбайта да бъде на разположение на потребителя.

Захранване

Това е най-интересният блок от Poqet PC. Джобният компютър се захранва с две (!) обикновени алкални

батерии AA (с размерите на нашите батерии R20 по БДС).Те са достатъчни за 100-часова работа, което означава, че средната мощност на потребление не е повече от 60 mW – само по себе си забележително постижение, като се имат предвид обемът на паметта и дисплеят на течни кристали с висока разделителна способност. Когато батериите се изваждат за подмяна, автономността се поддържа от достатъчно мощен кондензатор, който осигурява захранването в продължение на 5 – 10 min.

По всяка вероятност най-съществената икономия е постигната от принципно новата конструкция на дисплея. Неговата консумация е намалена с един порядък – от 100 mW, както е при почти всички преносими (laptop) микрокомпютри, на 10 mW. За съжаление фирмата Poqet PC е все още в процес на патентоване на цялостния дизайн на захранването и засега го пази в тайна.

Частично загадката се прояснява от следните функционални подробности. Паметта естествено изисква да бъде захранвана непрекъснато, но външните рам карти си имат собствени литиеви батерии с ресурс, не по-малък от две години. BIOS поддържа останалите блокове в спящо състояние – в това число влиза и процесорът, ако по това време не изпълнява някаква програма. Всъщност дежурна е само схемата, която следи дали няма натиснат някой клавиш. През това време дисплеят остава включен, така че за потребителя остава скрит фактът, че процесорът не работи. Но не по-дълго от 2 min – след това и дисплеят се изключва. Работата се възобновява със специален червен клавиш, означен с I/O.

Външен вид, размери и цена

Poqet PC представлява плосък паралелепипед с размери 222 × 109 × × 23,5 mm – почти колкото една видеокасета и се отваря като кутия с водни бои. На капака е монтиран плосък дисплей от течни кристали с размери 173 × 69 mm (диагонал 186 mm) и без допълнително изкуствено осветление. На него могат да се извеждат 25 реда по 80 знака в текстов режим или графика с разделителна способност 640 × 200 пиксела, съвместима с контролерите CGA.

Клавиатурата има 77 клавиша с квадратна форма и размер приблизително 13 × 13 mm. При натискане създават тактилно усещане посредством пружинен механизъм.

Масата му е 450 g. Цената 2000 долара.

Периферия (скромна)

Фирмата-производител предлага допълнително външно дискетно устройство за 3,5-инчови дискети (1,44 Мбайта) със захранване от батерии. Консумацията осигурява автономност от порядъка на 20 часа.

Poqet PC има един 80-игленкуплунг за връзка с IBM PC /XT (XT 80-pin card connector), по който могат да се предават данни със скорост до 19 200 бита/s в сериен режим RS232C. Специалният кабел за него има два куплунга за серийни портове – DB-9 и DB-25.

Фирмата разработва четец на карти с ром и рам памет за директен обмен на данни с паметта на IBM PC. Той трябваше да се появи на пазара към края на 1989 година, но до момента на предаването на броя в печатницата (11 декември) нямаме данни дали обещанието е изпълнено. Цената на бъдещото изделие е определена само приблизително – от порядъка на неколкостотин долара.

Оракулски бележки

Дали ще има бъдеще това малко чудо на техниката?

Нека да видим какви са му шансовете. На първо време Poqet PC ще го купуват очевидно все хора от средна ръка – търговски пътници, застрахователни агенти, менажери и бизнесмени. За тях цената е напълно приемлива – спомнете си, че толкова струваше една пълна конфигурация на IBM PC само преди две години.

В наистина джобно издание е събрана пълната мощност на 16-битов компютър от класата на IBM PC/XT, със съществено по-голямо бързодействие поради повишената тактова честота и голяма скорост на обмена на данни с външната памет. Засега единствената пречка е повишената цена на картите с външна памет.

Един Кбайт рам от тях струва приблизително 1 долар, което означава, че 4 картички с по 512 Кбайта памет ще струват колкото компютъра. И това е при максималния капацитет – при карти с по-малка памет цената за 1 Кбайт нараства до 1,5 долара.

Но за тези клиенти не трябват много програмни продукти – достатъчно е един добре подбран интегриран програмен пакет от класата на Framework III и две-три рам карти за данни, които могат да се прехвърлят върху дискети чрез настолен компютър. Останалите програми от първа необходимост са вградени в рома с обем 640 Кбайта – освен програмите, за които стана дума по-горе, е включена и една опростена версия на най-известната електронна таблица Лотус 1-2-3

Тъй като приложните програми ще се изпълняват директно от картите, разработчиците трябва да адаптират съществуващия софтуер. Но, както твърди производителят, в тази насока не се очакват съществени трудности. Всички програмни продукти, които са писани на езика Си и са компилирани на съвременните компилатори, ще изискват просто само едно ново компилиране на изходните текстове на програмите. Производителите на софтуер са доста заинтересовани от джобния компютър, защото при тези носители програмното пиратство ще бъде извънредно затруднено. По преценка на фирмата известните приложни програми, записани върху карти, ще струват с около 15 процента по-скъпо от програмите, разпространявани на дискети. Разумно е да се очаква, че ако обемът на производството се увеличи рязко, цената им също толкова рязко ще се намали, но аз мисля, че в обозримото бъдеще тя ще продължава да се различава от цената на дискетите поне с един порядък.

Сега тук е деликатният момент на прогнозата. По същество с Poqet PC се открива съвършенно нова ниша на световния пазар, която свежда изчислителната мощ на настолните микрокомпютри до размера на калкулаторите. А в тази област производителите от цял свят видяха възможността да се продадат няколко милиарда екземпляри, като цената им бе доведена до възможния минимум – колкото на пакет луксозни цигари. От тази гледна точка не вярвам тук да се образува вакуум.

Ако големите и малки производители видят същата перспектива в джобните компютри, както в калкулаторите, няма да минат и три години, когато заедно с читанката и учебника по аритметика първолачетата ще носят в чантите си и джобни компютърчета – може би с по-яки капаци.



Уважаеми читатели,
Отбележете в квадратчетата с 1, 2 или 3 материалите от броя, които класирате съответно на първо, второ и трето място. Ако по ваша преценка някой материал е маловажен или слабо разработен, можете да го отбележите с „Х".
В долните редове напишете какви теми, материали или въпроси бихте желали да прочетете в списанието.

ЕГН

Професия

СЪДЪРЖАНИЕ СОДЕРЖАНИЕ

	Стр.	
Драги читатели	1	Дорогие читатели
Новите вируси в България	2	Новые вирусы в Болгарии
За лудия и вирусните зеленчуци	6	О вирусах и компьютерной морали
Правен режим на програмните продукти	8	Правовая охрана программных продуктов
Епъл ЕКСПО	11	Эпл ЭКСПО
Пазете очите от дисплея	13	Берегите глаз от дисплея
КАД-система за жакардови тъкани	16	КАД-система для жакардовых тканей
МИКРОТЕКСТ II	17	МИКРОТЕКСТ II
Електронен ръкопис	22	Электронная рукопись
Как работят архиваторите	25	Как работают архиваторы
Отпечатване на екрана във файл	29	Запись экрана в файл
Програмиране на асемблер	30	Программирование на ассемблере
Напряко през байтовете	31	Напрямик через байты
Книгопис	34	Новые книги по информатике
Заочен конкурс по информатика	35	Заочный конкурс по информатике
Преброяване на битовете	40	Как пересчитывать биты
220 хитрини или какво може Bag of Tricks	24	220 трюков или что может Bag of Tricks
Клавиатурен драйвер	46	Драйвер для клавиатуры
Как да свирим акорди на Правец-8Д	48	Как исполнять аккорды на Правец-8Д
Джойстик за Правец-8Д	50	Джойстик для Правец-8Д
Тайната на безконечните ленти - 2	52	Секрет бесконечных лент - 2
Статии и програми за Правец-8Д	54	Статьи и программы для Правец-8Д
Макинтош IIci	56	Макинтош IIci
Късове от пита	58	Ломтики блина
Радикална полимеризация	59	Радикальная полимеризация
Меню за ДОС-8Д	61	Меню для ДОС-8Д
Джобният компютър POQET PC	62	Карманный компьютер POQET PC

Издание на ЦК на ДКМС

Главен редактор:
Георги Балански
87–09–14

1000 София
бул. „Толбухин" № 51 А
87-50-45

Дежурен редактор
Славчо Иванов

Дизайн
и техническа редакция
Люба Калпакчиева

Коректор
Златка Вълчева

За публикуване се приемат програми, записани на магнитен носител — дискета или касета, които редакцията връща обратно. Ръкописи не се връщат.

РЕДАКЦИОНЕН СЪВЕТ ст. н. с. инж. **Александър П. Александров**, акад. **Ангел Ангелов**, проф. **Ангел Писарев**, акад. **Благовест Сендов**, **Веселин Спиридонов**, доц. **Димитър Шишков**, инж. **Иван Марангозов**, ннж. **Иван Михайлов**, доц. к. т. н. **Лазар Лазаров**, **Николай Боев**, инж. **Пенчо Сираков**, чл.-кор. **Петър Кендеров**, инж. **Петър Петров**, ст. н. с. к. т. н. инж. **Пламен Вачков**, **Рашко Ангелинов**, инж. **Стефан Чернев**

Предадено за печат
13.12.1989 г.

Подписано за печат
20.02.1990 г.

Печатни коли 8

Формат 60/90/8

Тираж 26300

Цена 1,60 лв.

ДП „Д. Благоев" 2
София, ул. „Ракитин" 2
Телефон 43–431

Индекс 20 593

ПРИЕМНИ ДНИ
НА КОНСУЛТАНТИТЕ

По проблемите на Правец-8Д и компютрите от серията Правец-82

БОРИСЛАВ ЗАХАРИЕВ всяка първа и трета сряда на месеца от 14 до 16 часа.

По компютърните вируси и други проблеми, свързани с шестнайсетбитовите компютри

ВЕСЕЛИН БОНЧЕВ - всеки втори и четвърти вторник на месеца от 16 до 18 часа.

МикроТЕКСТ II

Макрокоманда	Код за стартиране	Действие
copy_text.mac	Ctrl – C – T	Избира текст и го копира на друго място
freeze_style.mac	Ctrl – F – S	Превръща форматирането с маски в директно форматиране
index.mac	Ctrl – I – W	Маркира всички думи от списък за включването им в азбучен указател
index_entry.mac	Ctrl – J – E	Маркира избрана дума или израз за включването ù в азбучен указател
move_text.mac	Ctrl – M – T	Избира текста и го премества
next_page.mac	Ctrl – J – N	Премества маркера в началото на следващата страница. Документът трябва да е предварително отпечатан или страниран
prev_page.mac	Ctrl – J – P	Премества маркера в началото на предишната страница. Документът трябва да е предварително отпечатан или страниран
repaginate.mac	Ctrl – R – R	Премахва зададените граници между страниците и странира наново, като за всяка страница очаква потвърждение. Не бива да се използва за документи с повече раздели
replace_with_glossary.mac	Ctrl – R – G	Замества предварително указан текст с текст от речника
replace_with_scrap.mac	Ctrl – R – P	Замества предварително указан текст с текста, съдържащ се в буфера
save_ascii.mac	Ctrl – S – A	Записва документа като неформатиран текстов файл. В командата Печат Общи трябва да е зададен драйвер за принтер PLAIN.PRD
save_selection.mac	Ctrl – S – S	Записва избрания текст във файл
table.mac	Ctrl – T – T	Поставя табулаторите в таблица след задаване на стъпката на първия табулатор и дистанцията между следващите
tabs1.mac	Ctrl – T – 1	Служи за бързо създаване на таблици по зададена стъпка и тип на табулатора
tabs2.mac	Ctrl – T – 2	Създава празна таблица по предварително зададен брой на колоните и отстояние на първата от лявата текстова граница
toc_entry.mac	Ctrl – T – E	Маркира заглавия за създаване на съдържание



КАД системата
за проектиране
на жакардови тъкани
автоматизира изцяло
всички етапи
от създаването
на жакардови тъкани.
За входни данни
в системата
служи рисунка
в условни цветове,
а като изход
се получава файл
за перфориране
на лента
за жакардов стан
или за програмиране
на епром.
Системата
се произвежда
от ДФ „ТЕСКОМ“
под търговската марка
ЛОГИТРОН 2000. СМ. 01.