КОМПЮТЪР ЗА ВАС

Издание на ЦК на ДКМС

2–3'86 Година втора Цена 1,20 лв.

ISSN 0205-1893

СПИСВА РЕДАКЦИЯТА НА ВЕСТНИК ОРБИТА

Сканиране и обработка:

Антон Оруш

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

ФОРУМ САНДЪЦИТЕ

МУЛТИПАК съчетава в едно професионална ***текстообработка***; ***структуриране*** на документи; ***таблична обработка*** на данни; ***релационна база данни***, съвместима с бази данни от типа на dBASE-II и dBASE-III; опростен, но много ефективен генератор на ***графика***; пълен набор от ***телекомуникационни възможности*** и изключително мощен ***процедурен език***.

Текстообработката на **МУЛТИПАК** прави възможно използването на различни видове шрифтове (<u>подчертан</u>, **удебелен** и с *наклон*) и свободното ***форматиране*** на целия ***текст*** или части от него. Налице е и изключително комфортен механизъм за обработка на писма и формуляри.

Базите данни разрешават релационно представяне на информацията. Улеснено е сортирането на записите във възходящ или низходящ ред по избрано поле, или последователно по няколко полета. Възможно е търсене по предварително зададени прости или сложни условия.

Таблиците са изключително подходяща форма за представяне и обработване на цифрова информация. Налице са различни формати за числовите данни – търговски, валутен, чрез проценти, чрез експонента и др. При изчисленията може да се използва широк набор от аритметически операции, логически отношения и специализирани статистически, финансови и други формули.

МУЛТИПАК предоставя и ефективен механизъм за ***графично изобразяване*** на данните във вид на стълбовидна, наложена, точкова, линейна, процентна (кръгова) или аналитична графика. Възможно е ръчното надписване и оразмеряване на графиките, добавяне на една графика към друга и получаването на цветни изображения.

МУЛТИПАК е разработен в ***Института по Техническа Кибернетика и Роботика*** – БАН.

Продуктът се разпространява от: ***СК "Национален Програмен и Проектен Фонд"*** и ***Комбинат по Микропроцесорна Техника – гр.Правец.***

(Горният текст е подготвен чрез **МУЛТИПАК** и е разпечатан на лазерен принтер.)

ДРАГИ ЧИТАТЕЛИ,

Този сдвоен брой на списанието предвиждаме да бъде в ръцете ви в навечерието на поредното голямо събитие в живота на нашата партия и общества – Тринадесетия конгрес на БКП. Сега, когато се прави равносметка на изминатия между двата конгреса път в социалистическото изграждане, за участието на всеки в изпълнението на партийните предначертания, ни се иска да хвърлим по-обстоен поглед на някои от задачите, които поставиха февруарският (1965 г) и Януарският (1986 г) пленуми на ЦК на БКП пред младото поколение. Да направим етапна оценка за изпълнението на стратегическата повеля за масово компютърно огромотяване на младежта и подготовката ѝ за прехода към третото хилядолетие. Примери за новаторски подход потърсихме в специализираната внедрителска организация на ЦК на ДКМС ИВСД "Авангард", с успехите на която ви запознава нейният директор инж. Иван Михайлов. За обучението на младежта за работа с електронноизчислителна техника, за популярната фамилия минироботи Робко , разказва заместник-директорът на Института за техническа кибернетика и роботика при БАН ст.н.с. инж. Недко Шиваров.

На партийния конгрес посвещаваме и конкурса, който обявяваме съвместно с ЦК на ДКМС и СО "Програмни продукти и системи" за създаване на нови, по-съвършени български програмни продукти. Досегашният ни опит в подобни конкурси показва, че страната ни разполага със значителен творчески потенциал от висококвалифицирани програмисти, способни да решат поставените в проекта за Тезиси на XIII конгрес задачи.

Вярваме, че читателите ни ще прочетат с удоволствие и статията на изтъкнатия съветски учен и пръв в света академик по информатика Андрей Ершов "За човешкия и естетическия фактор в програмирането".

При композирането на броя сме търсили хармонична равновесие между материалите, посветени на основите на програмирането, приложните програми и статиите с общообразователен и популярен характер. Отпечатването на сдвоен брой ни позволи да поместим и някои ценни, на по-дълги програми, като разработката на инж. Павел Киров, както и да постигнем по-голямо тематично разнообразие.

Тук полагаме основите на някои нови тематични направления и рубрики. На първо място ще посочим поредицата "Програмиране на машинен език и Асемблер". Ако у нас все още има изключително силен глад за литература по микрокомпютри, то той е най-остър именно при програмирането на ниско равнище, без познаването на което не е възможно майсторски и пълноценно да се използуват възможностите на компютъра.

Колкото да авторите, Орлин Вълчев е вече добре познат като съавтор на книгата "Ключ за компютър" и автор на поредицата "Дискова операционна система" във вестник "Направи сам". Борислав Захариев, студент във ВМЕИ "В.И.Ленин", е сътрудник на списанието от неговото основаване и "злите езици" твърдят, че пишел на машинен код едва ли не с бързината на професионална машинописка. Авторът на поредицата "Управляващ програма Монитор" инж. Димитър Евстатиев също е проверен като автор на "Компютърна азбука", публикувана в десет броя на "Направи сам" през 1984/85 година.

Вярваме, че ще посрещнете с интерес новата рубрика за дългоочаквания домашен компютър "Правец 8Д". Имаме уверението на ръководството на НПК по микропроцесорна техника гр. Правец, че първите компютри ще се появят на пазара в края на месец март! Поканили сме за автор един от най-добрите у нас специалисти по този вид компютри Димитър Вавов. Не може да не споменем и автора на поредицата "Пролог" Кънчо Иванов, преподавател в Националния учебен център към СО "Програмни продукти и системи", когото познаваме и от статиите му по информатика в списание "Наука и техника за младежта". Ще добавим, че Пролог вероятно ще стане един от най-разпространените езици и именно на него са се спрели японците в плановете си за създаване на петото поколение компютри.

Известна новост в броя е и задълбаването в хардуера на компютъра, за което сме дали думата на други двама изтъкнати специалисти от "родилния дом" на фамилията Правец – Направление "Персонални компютри" към ИТКР при БАН – инж. Петър Петров и инж. Борис Вачков.

Да спрем дотук с представянето на авторите, защото мястото не ни позволява да споменем всички. И така достатъчно ясно се открояват признакът, по който сме ги подбирали – да са измежду най-добрите специалисти и популяризатори в съответната област. Убедени сме, че именно това е най-верният път за непрестанно издигане на равнището на списанието. Конкурсната система за подбора на сътрудниците при нас е постоянно-действуваща и вратите на редакцията ни са широко отворени за всички, които желаят на равна нога да се включат в нея.

Инж. ГЕОРГИ БАЛАНСКИ

УЧЕБНА РОБОТИКА

НЕДКО ШИВАРОВ
ст. н. с. зам.-директор на ИТКР, зам.-председател на СИСА по техническия прогрес

Днес изчислителната техника и особено персоналните компютри (ПК) навлизат във всички сфери на човешката дейност. Разработените в ИТКР — БАН осемразредни ПК се произвеждат серийно в Комбината по микропроцесорна техника — Правец, вече три години. Имко—1, Имко—2, Правец—82, Правец—83 и Правец—8М са сега главните технически средства на втората компютърна грамотност. Внедрени са в производство и първите шестнадесетразредии ПК — Правец—16 и Мик—16.

Роботиката е едно от най-перспективните направления на техническия прогрес. Гъвкавите автоматизирани производствеии системи (ГАПС) изменят облика на производствените процеси. Но за да се запознае младежта с тази нова техника, да се подготвят хора, познаващи нейните възможности, са необходими иови, специални технически средства.

Фамилията Робко от учебии роботи и средства за изграждане на учебни ГАПС, разработваиа в ИТКР — БАН, цели заедно с втората компютърна грамотност нашата младеж да се огрaмоти по роботика и ГАПС. Персоналните компютри и учебната роботика се допълват взаимно много удачно. Важно е още от началото на обучението си младежта дз не остзне с убеждението, че компютърът е средство само за изчисления или обработка на текстова информация. Необходимо е да се демонстрират възможностите на компютрите да управляват различни машинн, да проектират детайли, да управляват цели производствени системи и процеси, както това стаза на практика а съвременното производство.

Учебните роботи и средства за изграждане на учебни ГАПС — Робко, се управляват и програмират с осемразрелните персонални компютри от фамилията Правец. При работа с тях обучаемите ще имат възможиост да се запознаят с прииципите на компютърното проектиране в най-елементарнзта му форма, ще получат познания по механика, задвижвания, сензори, изграждане на системи, програмиране и управлеиие иа машини и ГАПС. Фамилията Робко дава възможност постепеино с нарастващата сложност и при активно участие на обучаемия, да се развиват способностите на младия човек, най-вече иеговото творческо мислене. Той сам създава системата, написва програмите за нейното управление, прави проби, ианася корекции и така проследява целия процес, използувайки нагледни, прости, безопасни и сравнително евтини учебни средства.

ФАМИЛИЯТА РОБКО

е предн всичко набор от учебии минироботи — опростени и стилизирани модели на най-разпространените в света промишлени роботи, както и периферни устройства, действуващи модели на обработващи машини, робокари, автоматизираии учебии складове. Всички те се управляват от ПК.

Учебният минироботРобко—01 има пет степени иа свобода и може да манипулира предмети с тегло до 250 грама, като ги позиционира с повторяемост, по-добра от 0,1 мм. Той е снабден с богат набор от периферни устройства: въртяща маса — Робко-01М, учебен конвейер — Робко—01К, сензорен хващач — Робко—01СХ, магнитен хващач — Робко—01МХ, блок сензори — Робко—01С и др. С тези периферни устройства, управляваии от ПК, могат да се изграждат разнообразни системи.

Учебният робот Робко—10 има механичната конструкция на Робко—01, но е с микропроцесорно управляващо устройство. Програмира се н се обучава чрез бутонен пулт за управление. Може дз бъде свързеи към ПК и да работи с всички учебни периферни устройства Робко.

Учебният робот Робко—111 е стилизиран модел на най-разпространения в света робот — СКАРА (съчленена механична ръка). Може да работи с всички периферни устройства и се управлява от ПК.

Разработват се и предстои да се внедрят Робко—11 — учебеи робот, работещ в цилиндрична координатна система, и порталният Робко—100, работещ в правоъгълна координзтна система. Предвижда се и учебен робот, работещ в сферична координатиа система.

Учебният робот Робко—9 е мобилеи робот с вграден микрокомпютър. Той има 9 степени на свобода и се придаижва с триколесна система с акумулаторно захранване. Ръката на робота може да манипулира предмети с тегло до 250 г. Прнтежава възможности, по-важии от които са синтез на говор от фоиеми, вграден програмируем часовиик, сензори, разпознаващи 256 нива на осветеност и 256 ниаа на звук, измерване на разстояния до предмети, заобикаляне на препятствия н още миого възможности за работа нз напреднали ученици и студенти. Чрез жична или радиовръзка той може да се свърже към ПК и да бъде програмиран от иего. В автономен режим се програмира чрез клавиатура и пулт за обучение.

Следващата група изделия от фамилията Робко са устройствата, с които се създават

УЧЕБНИТЕ ГАПС

Учебната фреза Робко—101 има три степени на свобода и се управлява от ПК. Оборотите на постоянотоковия двигател на роботиия инструмент могат да се

регулират. Учебният струг Робко—1101 се управлява от ПК и може да обработва детайли в учебния ГАПС. И при фрезата, и при струга се прилагат похватите на компютърното проектиране на изработвания детайл с управляващия ПК.

Учебният автоматизиран склад Робко—1011 по същество представлява умален, стилизиран трансманипулатор, при който могат да се обработват малогабаритни палети, със заготовки или с вече изработени от учебния ГАПС детайли. Той се управлява от ПК, а натоварва и разтоварва палетната станция Робко—1001, която се монтира между склада и обработващите машини.

Учебният робокар Робко—1000 има товароносимост 500 г и се програмира и се управлява от ПК. Той осигурява транспортирането на заготовки и детайли в учебните ГАПС. Фамилията Робко е в процес на развитие и постепенно внедряване в производство. За производител е определен Заводът за електромедицинска апаратура в София. Предвижда се създаване на специализирано предприятие за производство на изделия от фамилията Робко. През 1985 г. в ЗЕМА бяха произведени над 1200 Робко—01. Практическият внедрител на учебната роботика, който развива активна дейност за нейното популяризиране в младежките клубове „Компютър", е ИВСД „Авангард". В системата на МНП внедрител на учебната роботика и нейн популяризатор е „Учтехпром".

Учебната роботика направи първите си крачки. Сега е изключително важно да се преодолеят всички трудности и в разработката, и при внедряването в редовно производство. Не по-малко важно е и създаването на програми, учебно-методически средства, ръководства и учебници, подготовката на преподаватели и ръководители. ЦК на ДКМС направи много за популяризирането и усвояването на компютърната техника и учебната роботика сред младежта, в популяризацията на учебната роботика се ангажират и други обществени организации. Сериозно работят за усвояване на учебната роботика институтите на МНП, училищата, висшите учебни заведения. Учебни роботи се разработват и във ВМЕИ „Ленин".

За повишаване на ефективността от приложението на микрокомпютърната техника във всички сфери на народното стопанство, за създаване на по-съвършени български програмни продукти

ЦК НА ДКМС
СО"ПРОГРАМНИ ПРОДУКТИ И СИСТЕМИ"
СП. "КОМПЮТЪР ЗА ВАС"
В. "НАПРАВИ САМ"

ОБЯВЯВАТ

КОНКУРС

ЗА РАЗРАБОТВАНЕ НА СОБСТВЕНИ ПРОГРАМНИ ПРОДУКТИ В ОБЛАСТТА НА:

I - БАЗОВ СОФТУЕР ЗА 8- И 16-БИТОВИ МИКРОКОМПЮТРИ - ДОПЪЛНЕНИЯ И РАЗШИРЕНИЯ НА СЪЩЕСТВУВАЩИТЕ ОПЕРАЦИОННИ СИСТЕМИ, ТЕХНОЛОГИЧНИ СРЕДСТВА, ПОДПОМАГАЩИ ПРОГРАМИСТА В РАЗРАБОТВАНЕТО НА ПРОГРАМНИ ПРОДУКТИ, ГЕНЕРАТОРИ /ПРОГРАМИ, КОИТО ПРИ ЗАДАВАНЕ НА ПАРАМЕТРИТЕ, ГЕНЕРИРАТ КОНКРЕТНИ ПРИЛОЖНИ ПРОГРАМИ/, ПОМОЩНИ И ДРУГИ ПРОГРАМИ.

II - ПРИЛОЖЕН СОФТУЕР, ЗА РЕШАВАНЕ НА ЗАДАЧИ ОТ ВСИЧКИ ОБЛАСТИ НА НАРОДНОТО СТОПАНСТВО ЗА 8- И 16-БИТОВИ МИКРОКОМПЮТРИ.

III - ИГРИ ЗА 8- БИТОВИ МИКРОКОМПЮТРИ /ВКЛЮЧИТЕЛНО И ЗА ПРАВЕЦ 8Д/ С НАЦИОНАЛЕН, ХУМАНЕН И ВЪЗПИТАТЕЛЕН ХАРАКТЕР.

НАГРАДИ

РАЗДЕЛ I
I НАГРАДА - 5 000 ЛЕВА
II НАГРАДА - 3 000 ЛЕВА
III НАГРАДА - 2 000 ЛЕВА

РАЗДЕЛ II
I НАГРАДА - 3 000 ЛЕВА
II НАГРАДА - 2 000 ЛЕВА
III НАГРАДА - 1 000 ЛЕВА

РАЗДЕЛ III
I НАГРАДА - 1 000 ЛЕВА
II НАГРАДА - 700 ЛЕВА
III НАГРАДА - 500 ЛЕВА

КРАЕН СРОК НА КОНКУРСА - 30.IX.1986

РАЗРАБОТКИТЕ ЩЕ СЕ ПРИЕМАТ В РЕДАКЦИЯТА НА ДВЕТЕ ИЗДАНИЯ

СОФИЯ
БУЛ. ТОЛБУХИН 51
ТЕЛ. 87-50-45 И 87-09-14

ЗАКОНОМЕРНАТА ЕВОЛЮЦИЯ НА МЛАДЕЖКАТА ШКОЛА В ПРАВЕЦ

МАЯК ЗА ИНФОРМАТИЦИ

МАРИАНА ВЛАЙКОВА

- ОЧАРОВАТЕЛНОТО ПЕРСОНАЛНО НАШЕСТВИЕ
- ВТОРА ГРАМОТНОСТ ЗА ВТОРИТЕ ПРОСВЕТИТЕЛИ
- ДЕСАНТЪТ ОТ ВМЕИ В КОМПЮТЪРНИЯ ЕПИЦЕНТЪР
- ДОКУМЕНТ ЗА ПРАВОСПОСОБНОСТ

Компютърното нашествие първоначално ни стъписа, след това ни очарова, после ии направи подозрителни, докато най-накрая го приехме като нещо, което се разбира от само себе си. Компютрите завладяха и продължават да превземат какви лн не области на живота, като при това съвсем „нескромно" предявяват и претенциите си: вярно е, че с тях се улесняват много дейности, но също така възникват и куп нови потребности. Необходими са програмни продукти, квалнфицирани хора, нова нагласа в мисленето.

След решението иа Политбюро на ЦК на БКП за осигуряване на комплексни условия за обучение и работа на младежта с електронноизчнслителната техннка и поетото от Комсомола шефство над тази стратегическа задача за кратко време страната ни се осея с клубове „Компютър", в които нахлу младеж, жадна за съприкосновение с малките светещи екрани. Но какво се оказа положението с ръководителите иа тези клубове? Естествено те са хора с подходящо висше образование. Но към тях се предявяват и по-специални изисквания — за работата с прохождащите в компютърната грамотност ръководителнте на клубовете се нуждаят от педагогически и методнческн умения.

Ето как броеницата от навързанн една за друга потребностн доведе до създаването на постоянно действуващата школа в Правец — центъра на нашата мнкрокомпютърна промишленост. Първоначално з школата, родена в края на 1984 г., е било предадено да се обучават работещите в системата на техническото и научното творчество на младежта.

Всъщност всичко така е н започнало. Курсовете по компютърно обучение с продължителност седмица, две илн месец са провеждани със заместник-дпректори на окръжните специалнзпранн клубове „Компютър", с работници от общински н окръжнн комсомолски комитети. Пред персоналните компютри в Правец са се учнли млади служители на Българската асоцнация за туризъм н отдих, учители от Странджа-Сакар, младн научнн работници, ученнци. През миналото лято там е функционирала иационална средношколска лагер-школа. В нея са се обучавали 180 първенци в олнмпиадите по математика и програмиране.

От откриването иа школата до сега преподаватели от СУ „Климент Охрндскн", ВАН, ВМЕИ „В. И. Ленин" и специалисти от различни институти са обучили около 500 души.

Но това бе само началото — от тази година обаче се прави своеобразен скок. ЦК на Комсомола заедно с ВМЕИ „Лепин" организира Нацноиална постоянно действуваща младежка школа „Компютър".

Какво е качествено иовото, различието? ВМЕИ „В. И. Ленин" ще осигурява лектори за учебния процес, като ги подбира не само измежду свонте кадри, но ще кани специалисти и от други научни звена. Във висшия институт ще се разработват учебните планове и програмните продукти, с негово съдействие ще се набавят методически и учебнн пособия, учебнотехнически средства, а също и ще се разработи наредба за приемане на курсистите и статут на документа, който ще им бъде издаван а края на обученнето.

Интересно е и друго. В школата ще се организират предимно два вида курсове: опреснителни с продължителност 45 днн и дългосрочни специализации с продължителност от два месеца до една година. В момента в тази нова форма на следдипломна квалификация се обучават около 40 ръководители на окръжни специалнзирани клубове „Компютър". Седемдесет дни, толкова е определеното за тях прнсъствено време в Правец. Междувременно курсистите ще полагат напитн, а в края ще представят н дпплом-на работа.

А ето и част от учебното съдържание. Младите специалисти ще разширят познанията си върху операционнпте системи, езиците на програмиране, алгоритмите. Те ще се запознаят също с българските персонални компютри, с българската микропроцесорна фамилия СМ 600, с Асемблера за

микропроцесорите СМ 601 и 6502, ще овладяват работата с различни периферни устройства. Естествено е в школата да се разглеждат и специализирани програмни средства и системи. В курса на обучение са включени още педагогика, методика и някои организационни, финансови и юридически проблеми от дейността на клубовете „Компютър".

Но това не е всичко за школата в Правец. Всяка свободна от заниманията по повишаване на квалификацията седмица ще се организира обучение на ученици от града и окръга, на работници в общинските и окръжните комсомолски комитети. А през свободните два месеца през лятото в нея ще се обучават най-надарените за информатиката ученици.

И още нещо ще укрепи авторитета на новата учебна единица. От тази година третата степен на обучение на някои специалности във БМЕИ „В. И. Ленин" ще се пренесе в Правец. Пак там ще се сформира и школа за следдипломна квалификация по микропроцесорна техника към същия институт.

Не са малко трудностите, които трябва да преодоляват организаторите от ЦК на ДКМС и от Общинския комитет на БКП в Правец, но резултатите са налице — Националната младежка школа „Компютър" доказа, че е най-авторитетният център за подготовка на младите хора у нас по информатика.

ЗАК
ДЕМИК ЕРШОВ

В края на декември 1984 г. бележитият съветски учен Андрей Петрович Ершов бе избран за академик на АН СССР към новосъздадената (март 1983 г.) Отделение по информатика, изчислителна техника и автоматизация. Той е първият академик по информатика в света.

Акад. Ершов е роден на 19 април 1931 г. в Москва. През 1954 г. завършва математика в Московския държавен университет и започва работа в Изчислителния център на АН СССР. Началото на неговия научен път е свързано с възникването и развитието на идеите и методите за транслация на езиците за програмиране. Тези методи и идеи залегнаха в основата на съвременното системно програмиране.

В края на 50-те и началото на 60-те години под ръководството на младия Ершов бе създаден Входен език — един от първите алгоритмични езици от високо ниво. В него бяха предложени редица нови конструкции — използуване на многомерни операции (матрично смятане), цикли без параметри и други, които бяха предшественици на някои понятия в съвременните езици за програмиране.

През 1960 г. А. П. Ершов преминава на работа в Сибирското отделение на АН СССР. Работите му откриха ново научно направление в дейността на отделението — изследвания и разработка на оптимизиращи транслатори за езици от високо ниво. Под негово ръководство беше разработена системата Алфа, с която бе решена задачата за създаването на транслатори, които да генерират висококачествени обектни програми. В Алфа-транслатора бе заложена теорията на Ершов за разпределение на паметта. Той предлага обща методика за оптимизираща транслация, включваща понятието смесена стратегия при декомпозиране на сложни конструкции на алгоритмичните езици, както и отделянето на междинно ниво — машинно-независим вътрешен език, необходим за целите на оптимизацията.

Организаторският опит при създаването на сложни програмни системи позволи на А. П. Ершов да разработи методи за организиране на програмистки колективи в рамките на голям проект, а също и да формулира общите принципи на програмирането.

За научната дейност на акад. Ершов са особено характерни широкият кръгозор и умението да се определи най-важното и перспективно научно направление. Когато развитието на изчислителната техника и необходимостта от най-пълно използуване на ресурсите на изчислителните системи поставиха задачата за създаване на системи за колективно използуване, А. П. Ершов стана инициатор на такива изследвания в СО АН СССР. Под негово ръководство бе създадена първата в СССР развита система за време-деление Аист-О, реализирана върху многопроцесорна изчислителна система. Редица архитектурни и програмни решения в тази система бяха използувани активно в следващите изследвания, свързани с изчислителните центрове за колективно ползуване.

Основните работи на А. П. Ершов напоследък са свързани с изясняването на фундаменталните основи на транслацията. Основополагащи са работите му по обобщаване на методите за транслация в случай на многоезикова транслация. По-натагъшно развитие на това направление даде предложената от Ершов концепция за смесени изчисления. Точното определение на смесените изчисления, коректността им, свързаните с тях преобразувания на различни класове схеми на програми — тези работи на Ершов дадоха методологична основа на перспективния трансформационен подход в програмирането.

През 1985 г. А. П. Ершов е избран за член на Асоциацията по изчислителна техника на САЩ, през 1970 г. става член-кореспондент на АН СССР, а през 1974 г. е удостоен с почетното звание „Заслужил член на Британското изчислително общество".

Акад. Ершов развива голяма педагогическа дейност. Той е професор по информатика на Новосибирския държавен университет от 1969 година. От няколко години е организатор и ръководител на изследванията по училищна информатика в ИЦ на СО АН СССР. Заедно със сътрудниците си създаде и ръководи раздела по програмиране в популярното съветско ученическо списание „Квант".

Акад. Ершов е голям приятел на българските информатици. Той е идвал многократно в България. Участвуваше активно в съвместната работа на БАН и ИЦ на СО на АН СССР.

За българските информатици академик Ершов е скъп и близък. Ето защо те го поздравяват за петдесет и петата му годишнина и му честитят академичното звание.

Честито, академик Ершов! Винаги добре дошъл на българска земя!

Доц. ДИМИТЪР П. ШИШКОВ секретар на Националния семинар по информатика към СМБ

Някому може да се стори странно намерението да се публикува в научно списание статия по толкова субективен въпрос. Историята на науката обаче показва, че в определени периоди естетическите, организационните и изобщо външните по отношение на техническото съдържание на някаква научна дисциплина фактори са имали понякога решаващ принос за формирането и развитието на дадената дисциплина. Сега, когато програмирането като наука и като професия встъпва в периода на своето самоопределение, анализът на човешките фактори в програмирането изглежда актуален за автора.

Работата е там, че независимо от, а може би и благодарение на признаването на програмирането за ключов проблем, сега за програмистите настъпват трудни времена. Вълната им армия постепенно попада в „плен" на администраторите и ръководителите, които се стремят да направят техния труд планируем, измеряем, еднороден и обезличен.

В читателя ще трябва да се създава впечатление, че авторът смята неправилна тази тейдеиция. Недостатъчната ефективност на труда на програмиста е, може би, главната причина за несъответствието между потребностите и възможностите за успешно прилагане на ЕИМ.

Така че подчиняването на програмирането на промишлените методи на работа е неизбежно. Но тази тенденция трябва да бъде балансирана от насрещната инициатива на програмиста. Той трябва да намери някаква система от вътрешни ценности в своята работа, която да му позволи по-лесно да асимилира индустриалните методи на работа, а където трябва — и да ги преодолява.

Авторът е убеден, че тази система от ценности съществува обективно, но ие е осъната докрай, не е известна на всички и поради това изисква разпространение и защита. Тя има много компоненти, като най-важният от тях, изглежда, е професионалният статут на програмиста (тук към програмистите се причисляват и системните аналитици), ио на автора му се иска да говори повече за естетическата, за емоционалната страна на програмирането, при това не толкова за

ЗА ЧОВЕШКИЯ И ЕСТЕТИЧЕСКИЯ ФАКТОР В ПРОГРАМИРАНЕТО

В памет на Г. И. КОЖУХИН

Академик
АНДРЕЙ П. ЕРШОВ

това, което възнаграждава програмиста, когато той излиза със своя продукт пред потребителя, а за това, което съставлява неговата нравствена опора, когато остава насаме с програмата или машината.

Програмирането става масова професия. Трябва да се има предвид, че сега тя вероятно е най-трудната от всички масови професии, при това за съжаление тази трудност не се признава достатъчно.

Трудността е, че именно програмистите непосредствено достигат пределите на човешкото познание, опирайки до алгоритмично неразрешими проблеми и до дълбоките тайни на работата на главния мозък.

Трудността е в това, че дълбочината на собствения стек на програмиста не бива да бъде 5—6 позиции, както според психолозите е при средния човек, а да е равна на дълбочината на стека, съответен на поредната задача, плюс още 2—3 позиции.

Трудността е също и в това, че програмистът трябва да притежава способността на първокласен математик към абстракция и логическо мислене в съчетаание с едисоновски талант да гради всичко от нули и единици. Той трябва да съчетава акуратността на счетоводителя с проницателността на разузнавача, фантазията на автора на криминални романи с трезвата практичност на икономиста. А освен това той трябва да има вкус към колективна работа, да разбира интересите на потребителя и много други неща.

В работата тая трудност може да бъде преодоляна само чрез голямо емоционално напрежение, което изисква от програмиста особено самосъзнание и вътрешна позитивна нагласа. Раебирането на тази нагласа е необходимо за ръководителите на програмистите и особено за възпитателите и учителите им. Като пример могат да се изброят няколко трудни въпроса, които могат да бъдат решени правилно, само ако напълно се вземат под внимание обсъжданите фактори:

— Възможна и необходима ли е организация на разработката на софтуера на конвейер?

— Кой и защо е по-трудно да се намери за осъществяването на софтуерен проект — ръководител или изпълнител?

— Как да се съчетае елитността на системното програмиране с неговата масовост?

— Как да се възпитава програмистът — чрез мироглед (университет) или чрез професионални навици (технически институт)?

— Какво представляват индивидуалните способности в програмирането, специфични и необходими ли са те?

— Може ли и трябва ли да се отделя проектирането на голяма програма от нейното изготвяне?

Тези въпроси са част от общ проблем, поради което ще направим само частни коментари, като се опитаме да свържем постановката им с анализа на човешкия фактор в програмирането.

За конвейера. В известен смисъл конвейерът е дяволско изобретение. Издигайки продуктивността на небивало ниво, той същевременно превръща човека в придатък на машиината. Конвейерният метод в програмирането може или да ликвидира интелектуалния компоиеит в труда на програмиста, или да предизвика неврози поради противоречието между монотонността и трудността на работата. Представете си човек, който е длъжен 8 часа иа ден, 5 дни седмично, 50 седмици годишно да решава само кръстословици, и ще разберете какво е това програмист, който се специализира например в написването на редактиращи програми.

За ръководителите и изпълнителите. Не бързайте да сложите ръководителя на първо място, обяснявайки, че по определенне ръководител се иамира или създава по-трудно. Хайде да помислим защо много често ръководителят на проект предпочита да започне с млади специалисти, отколкото с хора със стаж над пет години? Не е ли, защото предпочитаме да използуваме чистия лист и пластичността на младия човек, а ие да преодоляваме пасивното съпротивление на позрелия и по-малко ясния за иас 33-годишен глава на семейство? Но това означава, че ние не умеем хармонично да развиваме професионалните достойства на изпълнителя, така че те да ие се снижават с възрастта и да бъдат полезни не само за ръководителя, но и за него самия и за бъдещите му иачалиици.

Елитността на програмистите изглежда очевидна за автора и е сериозно предизвикателство към човечеството като цяло. Може да се надяваме, че това предизвикателство ще бъде прието и преодоляно. Тази мисъл ще бъде разшифрована по-късно.

Мироглед и професионализъм. Същината на проблема е да се признае, че програмирането изисква от човека по-особеи поглед

⇒

* Статията се печата с незначителни съкращения.

върху света, неговите потребности и еволюцията му, особеиа морална подготовка за своя дълг. Програмистът е войник на научно-техническата революция и като такъа трябва да има революционно мисление.

Сега ние сме готови да формулираме централния тезис иа тази статия. Той се заключава в твърдението, че програмирането притежава богата, дълбока и своеобразна естетика, която лежи в основата на вътрешното отношение на програмиста към своята професия и която е източник на интелектуална сила, ярки притежавания и дълбоко удовлетворение. Корените на тази естетика лежат в творческата природа на програмирането, а трудностите му и в неговата обществена значимост.

Съвсем не е просто да се отдели естетическата същност на произволен вид професионална дейност. Тази същност се реализира в субективни категории и дълбоко се преплита с етичиия кодекс на професията, с нейното техническо съдържание и юридически статут. Ето защо изброяването тук на естетическите компоненти на програмирането ще има също субективен характер.

Творческата и конструктивната природа на програмираието не изисква особени доказателства. Авторът би искал да изкаже може би твърде спорната мисъл, че в своята творческа природа програмирането отива значително по-далеч от повечето други професии, като се приближава към математиката и писателския труд. В повечето професии, използвайки силите на природата, ине само „опитомяваме“ едии или други физически или биологически явления, без задължително да вникваме в същността им. В програмираието пък като че ли отиваме докрай. Едии от тезисите на съвременната теория иа познанието: знаем нещо, ако можем да го програмираме, ясно характеризира максимализма иа иашата професия.

Друг много важен естетически принцип на програмираието е изключителната взискателност към завършеността на продукта. Разбира се, това е характерно за много инженерни професии, но програмирането и тук отива по-далеч. На нивото иа индивидуалната работа винаги съществува поразителен контраст между почти и напълно завършена работа. Стопроцентовостта иа програмирането е източник на трудността му и същевременно иа иай-дълбоко удовлетворение от работещата програма.

Машината, изпълняваща програма, се държи разумно. В такъв кулминационен момент програмистът съзнава, че програмата му, получила самостоятелен живот, материално въплъщава интелектуалните му усилия, които вече стават общо достояние. Това тържество на интелекта е вероятно най-силната и най-специфичната страна на програмирането.

Добросъвестният програмист се отнася с машината както добрият жокей с коия си. Знаейки и добре разбирайки възможностите й, той никога не ще си позволи да компенсира мързела на мозъка с безгрижно изразходване на ресурсите й. Това чисто естетическо отношеине към работата е най-ефективният предпазител от лекомислената „песимизация“ на софтуера, която понякога унищожава ефективността от използуването на машината.

Друга част от естетическата същност иа програмирането образуват компонентите му, свързани с неговата социална фуикция. Създаването и разпространението иа софтуера твърде много напомнят резултата от появата на книгопечатането. Както книгите коидензират външния образ на света в очите на авторите им и позволяват да се възпроизвежда процесът на познаиието му, така и програмите и банките от дании натрупват ниформационния и операционния модел на света и позволяват не само да се възпроизвежда, ио и да се предсказва неговата еволюция, давайки по този начин нечувана аласт над природата.

Да си добър програмист сега е привилегия като на грамотния човек в XVI век, правилегия, която дава право на програмиста да очакав аналогичио признанне и уважение от обществото. За съжаленне очакваннята не вииаги се сбъдват. Трябва обаче да се отбележи, че за подобно признание е нужиа работа и от двете страни. В частност, програмистът трибва да следва едии етичен принцип, който има общ характер за всеки професиоиалист, но има специална интерпретация за програмиста. Има три варианта: работа заради работата, работа за пари и работа в името на цел.

В координатната система на програмиста първите два мотива стоят иа преден план, макар че в абсолютната координаиата система има значеиие само третият. Във връзка с тоа трябва винаги да се помни, че програмистът може да достигне пълиа хармония с обществото, единствено когато лоялиостта към целта, в достигането на която неговата програма е само част, стане негова вътрешна нагласа.

Като се говори за обществената функция на програмирането, не може да не се отбележи, че по пътя към осъществяването й лежи един нерешен технически проблем — осигуряването иа събирателния ефект от програмирането, проблем, който непременно трябва да се реши. Продължителното и стабилното използуване на продукта на програмистския труд ще окаже решаващо въздействие на професиоиалиото самосъзнание иа програмиста.

Сега авторът би искал да завърши обсъждането на по-раио изброените алтернативи и трудни проблеми.

За индивидуалните способности в програмирането. Необходим ни е образът иа идеалиия програмист. Разбира се, това ще бъде митична личност. Но кой е казал, че на нас не са ии иужии митове и приказки за програмисти? Всеки от нас трябва поне веднъж в живота си да види или поне да чуе за чудо-програмнст, от чиято програма ие може да се махне нито една команда или който пише хиляда команди на ден, или намира грешка при вероятност за иамиране едно към милион и т. н. Свойствено е за човека да търси орнентири и примери. Именно от тези позиции трябва да се решава спорът за прословутите „примадоин“ в екипите от програмисти. Да се обявяват такива програмисти за иежелателни е най-малкото съсгледство или завист към техните изключителии качества. Авторът има късмет да срещне в живота си няколко такива примадони в програмирането, които при цялата си индивидуалиост и даже екстравагантност даваха неоценим принос в работата на групата особено в трудни ситуации.

За отделянето на проектите на софтуера от изготвянето му. Изглежда, правилното решенне иа този въпрос е невъзможно без отчитане на човешкия фактор и естетическата потребност, която пречи да се занимаваш с чужди идеи или да не виждаш овеществяването на собствената си идея. Да дааш технически проект в чужди ръце е все едио да пращаш децата си в интериат.

В заключение да се върнем към тезиса за елитността на програмирането и за неговото бъдеще. Нашата апология, на пръв поглед, подчертаваше изключителния, особеиия характер иа програмирането и неговите пределни изисквания към човешките възможности. Именно тази взискателност и определя онова същото предизвикателство към човека, за което се говори по-горе. По

време ив престоя си в САЩ през 1970 г. авторът бе особено поразен от новите идеи на професорите Марвин Мински и Сеймур Пейпърт за обучаването на деца. Те обориха съществуващата представа, че децата се учат несъзнателно по методи на подражанието. Те доказаваг, че човек се научава на нещо, само когато в главата му се образува блоксхема на действие, оформят се подиограми и се прокарват информационни връзки.

Така човек неизмеримо ще усили своя интелект, ако направи част от своята натура способността да планира своите действия, да изработва общи правила и начии за приложенията им в конкретната ситуация, да организира тези правила в осъзната и изразима структура, с други думи — ако стане програмист.

Някога четенето и писането са били съдба за избраници. Сега в епохата на грамотността, за достигането на която бяха нужни 1000 години, ние отделяме нова избрана категория хора, които стават посредници между човечеството и информационния модел на света, скрит в машиите. Правейки изкуството програмиране общо достояние, ние ще се лишим от своята елитна изключителност пред лицето иа израсналото човечество. Не е ли това най-висшият естетически идеал за нашата професия?

За да се направи такъв скок, на човечеството ще му трябват много по-малко от хиляда години, но сега все още сме много далеч от това. Авторът е дълбоко убеден, че работата, с която се занимава програмистът, изисква и от неговите колеги и още повече от ръководителите му значително по-дълбоко разбиране на мотивите за изпълнение иа професионалния му дълг и перспективите на жизнения му път.

Ние формулирахме редица актуални проблеми, свързани с човешкия фактор в програмирането. Изглежда, че най-главната не беше назована. Поколенията от хора се изменят значително побавно от поколенията машини. Авторът би искал да попита своите колеги-ръководители знаят ли те как програмистът на възраст над петдесетте да стане не по-малко полезен, отколкото 30-годишният.Честно казано, у иас сега все още не е намерен иачин, как да се асимилират ветераните в съвременните условия на изменения и нестабилност и по този начии професията на програмиста да се направи пожизнена и даваща на човека комфортното усещане за обществена и професионална полезност.

САМОУЧИТЕЛ

ПЪРВИ СТЪПКИ В ЛОГИЧЕСКОТО ПРОГРАМИРАНЕ

КЪНЧО ИВАНОВ
кандидат на математическите науки

ПРОЛОГ

● **ИЗКУШЕНИЕ ЗА ЗАКОРАВЕЛИ ПРОГРАМИСТИ**

● **ЯПОНСКАТА ВРЪЗКА С МАРСИЛИЯ**

● **ЕЗИК ЗА МАТЕМАТИЧЕСКИ И ЖИТЕЙСКИ ТРИЪГЪЛНИЦИ**

През 1985 г. сдружението „Програмни продукти и системи — АВАНГАРД“ пусна в продажба езика за логическо програмиране Пролог, предназначен за персонален микрокомпютър (МК) Правец-82. В потока от множество програмни продукти, които иахлуха в нашата практика, споменатият език остана недостатъчно забелязан. Причините са няколко. Използуването на Пролог изисква разширение иа паметта на микрокомпютъра с 16 Кбайта. От друга страна, единствената документация към езика е *Ръководство за програмиста*. По същество това е справочник за езика и, разбира се, не може да служи като учебно пособие за начинаещи програмисти. Представената версия на Пролог е удачно замислена, което е едно от редките изключения сред разпространяваните у нас програмни продукти. Авторите са се постарали да разработят удобен за българските потребители вариант на езика. Програмирането на Пролог се извършва на български.

Друга причина за все още слабата му популярност сред българските програмисти се крие в оригиналната концепция на езика, която поне засега изглежде непривична за специалисть, закоравял в програмиране иа Бейсик, ПЛ/1 или Паскал. Програмата на Пролог е понятие, което предизвиква усмивка сред начинаещи програмисти, очаровани от възможностите си, след като са овладели по-голямата част на Бейсик и са започнали да надничат в програми, написани на Асемблер.

Има обаче редица съображения, които са причина за изостреното вниманне на специалистите, следящи отблизо тенденциите в развитието на езиците за програмиране. Преди всичко Пролог бе избран за основа иа разработките по програмио осигуряване в японския проект „Пето поколение компютри“. Наред с това съществено е, че първият вариант на езика е бил предложен от специалисти по *изкуствен интелект*, които са се заннмавали с разработка на *човеко-машинен интерфейс* на естествен (в случая френски) език. Езикът е създаден като резултат от усилията да се опростят средствата за взанмодействие с компютъра и да се направят по-приемливи за все по-разширяващата се маса от потребители. Важно е да се подчертае, че създателите на езика са вървели по нетрадиционен път. Те не са продължили

⇒

опитите по създаване на допълнителен комфорт за потребителя чрез подобряване на диалога с обогатени менюта, раздвижена графика, светлинни и звукови трикове.

Използван е съвсем нов подход на програмиране. За това, доколко неестествено за програмистите се оказа *логическото програмиране*, говори следният факт: Ален Колмероер създава в Марсилия през 1971 г. първата версия на Пролог. Необходими бяха 10 години непрекъснати усилия на ентусиазираните популяризатори на новия стил, докато избухне бомбата със закъснител. През 1981 г. японците издвусмислено заявиха — успехът на проекта за разработка на компютрите в последното десетилетие на нашия век ще зависи от развитието на идеите, заложени в Пролог.

За какви идеи става дума? Названието Пролог (PROLOG) — произлиза от PROgramming in LOGic, което означава логическо програмиране. Логическото програмиране е едно от направленията на *непроцедурното програмиране*. Стремежът на привържениците на непроцедурното програмиране е да премахнат „директивния" дух на програмирането, изразяващ се чрез команди за действия: READ, WRITE, GOTO, RETURN и т. н. На директивността се противопоставя описателността. Наистина всяка задача може да бъде описана.

„Страните на даден триъгълник са съответно a=15, b=18, а периметърът му P=50. Търси се дължината на страната c."

Естествено: c=17.

Наистина изглежда по-привлекателно да представим задачата в следната форма:

a=15
b=18
P=50
c=?

вместо да подготвяме алгоритъм и съответна програма за компютъра. Трябва да се има предвид, че при „описателния" подход е необходимо на компютъра да се дадат още знания. Очевидно той трябва „да знае", че:

P=a+b+c.

Но по-важното е, че той трябва да притежава механизъм за извличане на необходимия резултат от условието на задачата и допълнителните знания, които притежава.

Възникването и развитието на езиците за програмиране от високо ниво стана възможно благодарение на усилията на редица изтъкнати математици, между които трябва да отбележим Яи Лукашевич и Джон Бекъс. Те показаха, че съществуват алгоритми за автоматично преобразуване на традиционни математически формули в машинни команди. С други думи — намериха алгоритъм за изчисляване на произволна *правилно записана* формула. Извличането на резултат от дадени начални условия и знания е много по-сложна задача, чиито кореии са вплетени в благодатната тематика на *математическата логика*. За да научат компютрите да правят подобни изводи, специалистите по изкуствен интелект трябваше да намерят *алгоритъм за доказване на теореми*. Такъв алгоритъм съществува благодарение на постиженията на формалната логика, от една страна, и на предложената от Клод Шенън *минимаксна процедура*, която реализира *метода на пробите и грешките* при търсене на най-доброто решение измежду множество възможни. Първите реализации на подобни алгоритми действуваха отчайващо бавно. Ето защо изглеждаше наивно да се говори за системи, които ще правят изводи на основата на множество твърдения и факти. Но подкрепяни от редица подобрения на базовите алгоритми и от постижения на логиката като *унификацията* (предложена от Жак Ербран) и *принципа на резолюцията* (предложен от Дж. Робинсон), съвременните компютри „правят заключения" много по-бързо.

Постепенно непроцедурното програмиране набра сили и предизвика преоценка на архитектурата на съществуващите компютри — заговори се за *нефон-нойманoвски* архитектури. Всъщност Пролог е представител на едно от направленията в непроцедурното програмиране. Програмата на Пролог не представлява свободно описание на условия на решаваната задача и на необходимите знания. Истината е по-сложна. В програмата, написана на Пролог, се съдържат множество факти, представляващи *база от знания* (БЗ). В случая под факт се разбира релация (връзка, отношение) между няколко обекта, описана като предикат. Така например фактът, че ИВАН СЛУЖИ В ЦИИТ, се определя чрез предиката СЛУЖИ, след който в скоби се поместват ИВАН и ЦИИТ:

> СЛУЖИ (ИВАН, ЦИИТ).

> СЛУЖИ (ЛИЛИ, БТ).

Вторият предикат изразява факта, че Лили е служителка на Българската телевизия (БТ). По подобен начин могат да се изразят факти за телефоните, професиите, адресите и т. н. на определени лица например наши приятели. Съвкупността от знания представлява програма!! Върху базата знания могат да се поставят въпроси от страна на потребителя, на които Пролог може да дава отговор. Например: „Кой служи в БТ?" Оформянето на въпроса на Прилог изглежда така:

? — СЛУЖИ (R, БТ).

Обърнете внимание на факта, че търсената стойност е изразена във въпроса чрез променливата R.

Отговорът на компютъра е:

R=ЛИЛИ

Опитните програмисти веднага ще отбележат, че Пролог притежава възможности за търсене в база от данни. Наистина базата от знания се обслужва чрез механизма за достъп до релационна база от данни. Ето защо при известно натрупване на факти и усложняване на въпросите скоростта на работата на Пролог системата се понижава. Важна подробност — скоростта на системите за логическо програмиране се измерва в специални мерии единници — липсове. Липс (LIPS — Logical Inference Per Second) означава едни логически извод за секунда. Съвременните системи работят със скорост от порядъка на 10^4—10^5 липса. Създателите на петото поколение компютри се надяват техните системи да достигнат скорост 10^9 липса, при това разчитат да работят с бази от знания с обем от порядъка на 1 Тбайт (приблизително 1 милиард Кбайта). Наистина амбициозни замисли! Но да се върнем към конкретната система Пролог, с която разполагаме на микрокомпютъра Правец-82. Това е система с ограиичеии възможности по отношение на последните постижения в логическото програмиране. Както вече отбелязахме, скоростта ѝ не е много голяма. Но за начално запознаване с

особеностите на логическото програмиране може да служи напълно успешно.

След стартиране на системата (засега ще отложим разглеждането му) потребителят получава следната комбинация от символи на екрана:

?

— Това означаза, че потребителят може да задава въпроси. Както вече се досещате, режимите на работа са два: задаване на въпроси и внасяне на нови факти в базата от знания.

В самото начало базата от знания е празна, поради което обикновено тя се създава. Това се постига чрез внасяне на нови факти в паметта на компютъра. За целта трябва да се премине в режим *внасяне на знания*, признакът за който върху екрана е:

>

Ето защо ще различаваме два основни потребителски режима: *въпрос* (?—) и *внасяне на знания* (>). След стартирането на системата тя е в режим *въпрос*. За да премине в режим *внасяне на знания*, трябва да й отговорим с „УЧИ“, т. е. учи́ се на нови знания. Ето как изглежда диалогът:

?—УЧИ.

>

Знакът > показва, че системата е готова да приема новите знания, които ще й предложим. Всяко ново знание, което задаваме за включване в БЗ, завършва с точка.

СЛУЖИ (ИВАН, ЦИИТ).

СЛУЖИ (ЛИЛИ, ВТ).

СЛУЖИ (ПЕТЪР, КОРПОРАЦИЯ-ППС).

СЛУЖИ (СТОЯН, ИТКР-ВАН).

Въведените знания се съхраняват в БЗ, която се изгражда в паметта на компютъра. Към фактите от тази област могат да се наставят съответни въпроси. За да преминем от режим *внасяне на знания* към режим *въпрос*, се използува клавишът @. Ето как изглежда диалогът със системата.

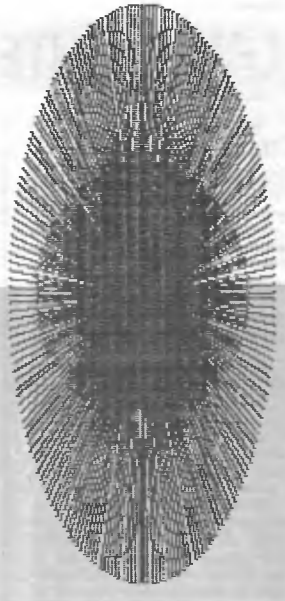
> @.

?—

Понякога при този диалог компютърът отговаря с предупредителен текст „ГРЕШКА31“, но това няма особено значение, тъй като в крайна сметка се преминава в режим *въпрос*.

СЛЕДВА

СУПЕРМИНИ-КОНКУРС „10—20“



```
0  REM  КИРИЛ ИВАНОВ
10  HGR2 : HCOLOR=
 3:XP =  - 135:YP
 =  - 75: FOR Z
 = 0 TO 6.28 STEP
 .03:X = 140 +
 XP *  COS (Z):Y
 = 96 + YP *  SIN
 (Z): HPLOT 140,96
 TO X,Y: NEXT
20  HCOLOR= 3:
 FOR Z = 0 TO 6.28
 STEP .04:X = 140
 + 60 *  SIN (Z):Y
 = 96 + 60 *  COS
 (Z): HPLOT 140,96
 TO X,Y: NEXT
```

```
0  REM  ИВАЙЛО ИЛИЕВ        СОФИЯ
10  DATA 162,248,160,4,138,72,234
      ,234,234,173,48,192,136,208,
      250,104,170,168,136,208,253,
      232,224,248,208,232,238,3,10
      ,238,3,10,174,3,10,224,16,20
      8,217,162,4,142,3,10,96,173,
      79,3,240,29,201,13,176,25,10
      ,10,10,160,4,174
11  FOR A = 2560 TO 2639: READ B:
      POKE A,B: NEXT : CALL 2560:
      DATA    128,3,72,189,32,192
      ,104,170,202,208,253,136,208
      ,241,56,233,4,176,234,96
```

```
0  REM  СОФИЯ:"АМИ
 БУЕ"#33:НИКОЛАЙ
 ХРИСТОВ
10  HOME : VTAB
 12:A$ = "
    <*> СУПЕРМИНИКОНКУРС
 '10-20' <*> ИНТЕРЕСНА
 ПРОГРАМА ЗА ПОКАЗВАНЕ
 НА ТЕКСТ ПРЕДЛАГА
 НИКОЛАЙ ХРИСТОВ
 СОФИЯ 1985"
15 R
20  HTAB 1: PRINT
  LEFT$ (A$,39):: A$
 =  MID$ (A$,2)
 +  LEFT$ (A$,1):T
 =  PEEK ( - 16384):
 IF T <  > 160
 THEN  FOR I =
 1 TO 30: NEXT
 : GOTO 20
```

```
0  REM  ПАВЕЛ ПЕЕВ
10  HOME :A$ =
 " СУПЕРМИНИКОНКУРС
 & ПАВЕЛ ПЕЕВ":M
 =  LEN (A$): FOR
 L = 1 TO M:H =
 (20 - M / 2) +
 L:L$ =  MID$ (A$,L,1):
 FOR V =  INT (
 RND (9) * 20)
 + 1 TO 1 STEP
  - 1
20  VTAB V: HTAB
 H: PRINT L$: CALL
  - 868: NEXT :
 PRINT  CHR$ (7):
 NEXT : FOR P =
 1 TO 1000: NEXT
 : RUN
```

ПРОФЕСИОНАЛНО КРЪЩЕНИЕ

АСЕМБЛЕР И МАШИНЕН ЕЗИК

ОРЛИН ВЪЛЧЕВ
БОРИСЛАВ ЗАХАРИЕВ

НЕОБХОДИМОСТТА ОТ НИСКОТО СТЪПАЛО

Съществуват много високоразвити езици — такива са Паскал, Алгол, Фортран, Бейсик, Кобол и др. Езикът Бейсик е език и ма микрокомпютрите Правец. С него започват и първите опити за програмиране. На Бейсик могат да се напишат иелошн програми с лесна проверка, поправка и изпълненне. Защо е необходимо тогава да се слиза на ниското стъпало — машиию-ориентираиите езици? Защото иа всяка програма на Бейсик може да се съпостави програма на Асемблер, но не всяка асемблерска програма може да се напише на Бейсик. Нещо повече, в някои случаи това е напълно невъзможно.

Отсега нататък под *асемблерска програма* ще се разбира програма, написана на езика Асемблер. Такава програма може да бъде изпълиена едва след транслация и зареждане в паметта на компютъра. Заредената в паметта програма се нарича *машинна програма*. Машинната програма може да бъде въведена директно в паметта на компютъра (чрез Моиитора) или да бъде заредена там след транслация (превод) от някакъв език за програмиране — Асемблер, Бейсик, Паскал и т. н.

Ето и ияколко довода в полза на програмирането иа Асемблер:

- **Времето за изпълненнето на асемблерска програма спрямо функционално аналогична програма на Бейсик е десетки, стотици, а а някои случаи и хиляди пъти по-малко!**
- **Паметта, заета от асемблерска програма, е също ио-малка — от няколко до десетки пъти.**
- **Винаги може да се оценят точно обемът на паметтта и времето за изпълнение.**
- **Програмирането на Асемблер или директно на машинен език позволява най-пълно използване на възможностите на компютъра.**

Посочените предимства не означават, че трябва да се отказваме от Бейсик. Оптималното решенне е следното: алгоритмично по-трудните участъци от дадена програма, които обаче ие са критични по отношение на време за изпълнение, да се пишат иа Бейсик. Подпрограми, изискващи минимално време за изпълненне или обслужващи специализирана периферия, се пишат на Асемблер.

ОБЩИ СВЕДЕНИЯ ЗА ПРАВЕЦ-82. РАЗПРЕДЕЛЕНИЕ НА ПАМЕТТА

Работата с езици от високо ниво ие изисква задължително познаване на структурата на компютъра. Програмнрането на Асемблер обаче изисква познания за архитектурата на конкретната система, за разпределението на паметта и за редица особености в конструкцията на компютъра, за които пишем програмата. Затова казваме, че Асемблерът е машинно-зависим език.

С други думи, асемблерската програма, написана за микропроцесора (МП) 6502 (процесорът на Правец-82 и Правец-8Д), не може да бъде изпълнена от друг микропроцесор. Поради специфичната оргаиизация на паметта при различните мнкрокомпютри машииа програма, иаписана за здин компютър, ие може в общия случай да бъде изпълнена върху друг компютър, дори той да е със същия микропроцесор. Както всички езнци, така и Асемблер притежава различни диалекти.

Оттук следва, че на ниво първнчен код (асемблерски команди и оператори) програмите трябва да са пригодени за траислатора от Асемблер, работещ върху съответния компютър, а така също и да са съобразени с уникалната за всеки компютър организация на паметта. Тук трябва да споменем и за често използваните системни подпрограми, чиито функции и адреси варират дори при различни варианти на здна и съща машина.

Такъв е случаят при Правец-82 и Правец-8М.

Паметта на Правец се състои от 65536 осемразрядни клетки (байта). Три четвърти от тази памет е тип *РАМ*, т. е. от нея може да се чете, а в нея — да се пише. Тук и по-иататък под *четене от паметта* ще разбираме извличане на информация от паметта, а под *запис в паметта* — поместване на данни на някой от валидните адреси. В *РАМ* се поместват потребителските машинни програми и данните за тях, програмите иа дисковата операционна система (ДОС), текстовите и графичните страници, както и някои специални области, свързани с работата на системата.

Остаиалата четвърт от адресируемата памет — 16 Кбайта, е разпределена иа две. Четири Кбайта от адресното пространство са определени за работа с пери-

ферин устройства (2 Кбайта са налични и 2 Кбайта са резервирани за евентуални *ROM*-разширения). Тук се намират и някои специални адреси, свързани с управлението на компютъра. В останалото 12 Кбайтово пространство са поместени интерпретаторът на Бейсик и управляващата програма Монитор. Тези два системни компонента са записани в интегрална схема тип *ЕПРОМ* (EPROM Erasable Programmable Read Only Memory), откъдето може само да се чете информация.

Тъй като за нас интерес ще представляват програмите на машинен език (въведени директно или транслирани асемблерски програми), ще обърнем внимание на свободните области от паметта, удобни за разполагането им.

а) областта от адрес $ 00 до адрес $1FF е забранена за машинни подпрограми;

б) адресното пространство от $200 до $2FF е входният буфер за клавиатурата. Ако сте сигурни, че няма да въвеждате максимално допустимите 256 символа наведнъж, то последните (старшите) адреси от тази област могат да се използват. Все пак записването на машинни програми в тази област не е за препоръчване;

в) $300 — $3BF

Удобно място за кратки подпрограми на машинен език;

г) $400 — $7FF

Използването е невъзможно, защото всяка команда, свързана с редактирането на текста от екрана, ще предизвика промени или унищожаването на програмата. Бейсик операторът HOME например ще изтрие цялата програма!

д) $800 — $ 1FFF

Възможно е поместването на машинна програма, ако тя заема мястото на програма на Бейсик. В нормалния случай — когато Бейсик програмата е къса и започва от стандартния си адрес $801, машинната програма може да се разположи след нея. Съществува възможност Бейсик програмата да се премести от по-висок начален адрес — например $6000. Тогава и тази област може да се смята за свободна;

е) $2000 — $5FFF

Възможно е разполагането на машинни програми, ако не се използват режимите с висока разделителна способност (HGR и HGR2). Ако се използва само първата графична страница (HGR) с адреси от $2000 до $3FFF, е възможно разполагането на програми от адрес $4000 нагоре;

ж) $6000 — $9000

Това е удобно място за поместване на машинни подпрограми. Трябва да се внимава да не бъдат препокрити променливите от тип низ (стрингови променливи) на Бейсик, които се разполагат от адрес HIMEM надолу. При зареден ДОС с три входно-изходни буфера HIMEM е $9600;

з) $9600 — $BFFF

Възможно е разполагане на програма, ако след зареждането ѝ няма да е необходим ДОС или в случая на работа без ДОС (например с касетофон);

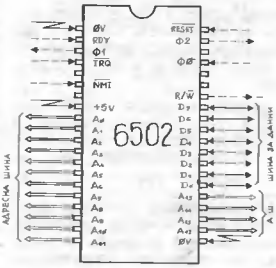
и) $C800 — $CFFF

$FFFF $F800	Монитор; Вектори на прекъсване
$F7FF $D000	Бейсик интерпретатор (в ЕПРОМ)
$CFFF $C800	Няма налична памет за потребителски програми (нито РАМ, нито ЕПРОМ)
$C7FF $C000	Входно-изходни адреси
$BFFF $9600	Дискова операционна система (ДОС), заредена само от дискета
$95FF $6000	Свободна област за програми и данни
$5FFF $4000	Втора графична страница (HGR2)
$3FFF $2000	Първа графична страница (HGR)
$1FFF $0800	Бейсик програма (нормално състояние)
$07FF $0400	Първа текстова страница
$03FF .... $ 03C0	Адреси за работата на ДОС и Монитор
$03BF $0300	Свободна памет за кратки машинни програми
$02FF $0200	Входен буфер (вход от клавиатурата)
$01FF $0100	Системен стек. Тук се записват: — адреси при преходи към подпрограми; — адреси при прекъсване на микропроцесора; — параметри и данни
$00FF $0000	Нулева страница. Съдържа адреси и константи за работа на системата.

Машиниата програма може да се разположи иа тези адреси само ако е записана в епром;

к) $D000 — $FFFF

Пространството е неизползваемо!



МИКРОПРОЦЕСОР 6502

Микропроцесорът 6502 е голяма интегрална схема с висока степен иа интеграция, поместена в едии корпус с 40 извода (*фиг. 1*). Изпълнен е по NMOS технология. Ето и предназначението на някои от изводите:

— 16-битова магистрала за адрес (адресни шии). Формираният от микропроцесора 16-битов адрес се подава към паметта или към периферните устройства. Тези шестнайсет бита определят адресно пространство от 64 Кбайта (от $0000 до $FFFF);

— 8-битова двупосочна магистрала за данни. По тези шини се предава информация от микропроцесора към паметта или периферните устройства и обратно;

— управляващи шини. Такива шини са RESET, IRQ, NMI, тактовите поредици Ф1 и Ф2, захранването иа микропроцесора и др.

Работата иа микропроцесора е строго разчетена във времето от двете незастъпващи се тактови поредици Ф1 и Ф2. Те се наработват от външен генератор (често наричан часоаник). През такта Ф1 микропроцесорът изработва адрес към паметта или извършва свои вътрешни операции. По време на такта Ф2 се обменят данни с паметта или периферията (запис-четене).

Шиията RESET служи за установяване на микропроцесора в иачално състояние. Начинът, по който става това, ще бъде изложен по-нататък.

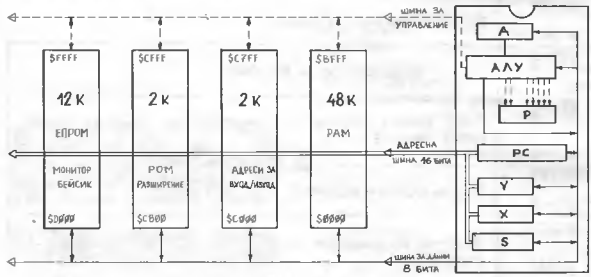
Шините IRQ и NMI служат за прекъсване на работата на микропроцесора. Програмистът може да забранява прекъсването по IRQ шината.

При обмена иа дании с паметта микропроцесорът изработва и управляващия сигнал R/W, който показва посоката — запис или четене.

Част от структурата на микропроцесора са регистрите. Те наподобяват клетки от паметта с тази разлика, че избирането им става не чрез формиране иа адрес, а по програмен път. Не всички регистри в микропроцесора са програмно достъпни. Важна особеност на регистрите е възможността за извършване иа определени аритметични и логически операции в тях. При тези операции важна роля играе аритметично-логическото устройство (АЛУ) в микропроцесора.

Програмистът иа Асемблер трябва добре да познава програмно достъпните регистри и техните възможности.

МП 6502 притежава следните достъпии за програмиста регистри *(виж фиг. 2):*



А — 8-битов акумулатор. Това е един от най-важните регистри. Има непосредствена връзка с аритметично-логическото устройство, което управлява извършването на различните аритметични и логически операции върху съдържанието на акумулатора или между акумулатора и клетка от паметта. Обикновено резултатът от операцията остава в акумулатора А;

Х, У — два 8-битовн регистъра. Наричат се индексни регистри. Участват при определяне на адреса на операндите, служат за организиране иа броячи, съхраняват времеиио данни и т. н.;

PC (PROGRAM COUNTER) — специален 16-битов регистър. Съдържа адреса на командата, която се изпълнява от микропроцесора. Нарича се още програмен брояч или брояч на адреси;

S — 10-битов регистър — указател към стека (съдържа адрес на клетка от стека). Програмно достъпни са само младшите 8 бита, което предполага максималиа дължина на стека 256 байта. В старшите 2 бита е запнсано постоянно 01, което фиксира стека от адрес $100 до $1FF. В стека (област от оперативната памет) микропроцесорът запиева адреси на връщане при преход към подпрограми или при прекъсване иа работата му. Стекът може да служи и за предаване на стойности на параметри между програми;

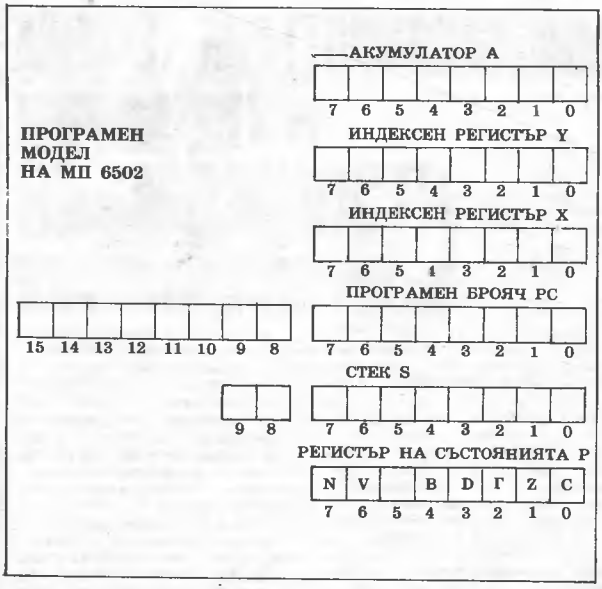
Р — 8-битов регистър на състоянията. Седем от 8-те бита служат като флагове за резултата от последиата операция или като флагове, управляващи работата на микропроцесора. Този регистър се използва от програмиста при така наречеиите условни преходи (преходи, зависещи от настъпването иа иякакво събитие).

Ето значеннето на отделиите битове в регистъра на състоянието (Р):

БИТ	ЗНАЧЕНИЕ	МНЕМОНИЧЕН КОД
0	Пренос	C(Carry)
1	Нулев резултат	Z (Zero)
2	Забрана на прекъсването	I (Interrupt)
3	Десетичен режим	D (Decimal)
4	Програмно прекъсване	B (Break)
5	Не се използва	
6	Препълване	V (oVerflow)
7	Отрицателен резултат	N (Negative)

МП 6502 разпознава 56 различни кодове от операции и има 13 вида адресации (начини за формиране на адрес от паметта). Машинният цикъл е приблизително 1 микросекуида.

ерът изпълнява съответната операция. Тя може да изисква обръщение към паметта. В такъв случай микропроцесорът прочита от паметта един или два байта, разположени непосредствено



Под програмен модел на микропроцесор *(фиг. 3)* ще разбираме набор от достъпните на програмиста регистри, представени с техните мнемонични означения (кодове). Тези означения се използват при писане на програмн на Асемблер.

Микропроцесорът се отличава от останалите интегрални схеми и по това, че има свой собствен език — набор от команди, които микропроцесорът разпознава и изпълнява. Кодът на операцията (КОП) се записва в една клетка (един байт) от паметта. Не всички 256 възможни стойности са допустими КОП.

Как работи микропроцесорът? Съдържанието на програмиия брояч определя един адрес, от който микропроцесорът прочита здин байт. Прочитането става по шните за данни, а съдържанието на клетката се разполага в специален, програмно нздостъпен регистър, наречен *регистър на командите*. Съдържанието на регистъра се анализнра от микропроцесора за иаличне на допустима комбннация от иули и единици, т. е. за съществуващ КОП. Ако прочетеният байт е допустим, т. е. КОП, микропроцесо-

след байта с кода. На базата на тази ннформация микропроцесорът формира адреса на клетка, в която се намира операндът (стойността, необходима при изпълнение на исканата операция, аргументът). Типът адресация определя начина, по който ще се стигне до операнда — той може да следва непосредствено КОП в паметта, да бъде на адрес, който следва КОП или пък адресът му да се формира и чрез някой от иидексните регистри. Възможно е разпознатият КОП да не изисква обръщенне към паметта.

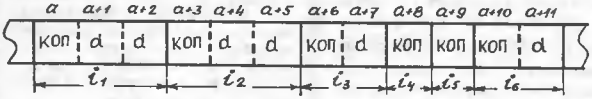
Под команда ще разбираме кода на операцията плюс допълнителната ннформация, необходима за откриване на операнда. Очевидно ще има команди, състоящи се само от КОП — опэрации само над регистри.

След разпознаването на КОП микропроцесорът увеличава брояча на адресите (програмния брояч) с 1, 2 и 3 в зависимост от дължината на разпознатата команда. Така броячът сочи адреса на следващата за изпълнение инструкция. КОП отново се прочита в регистъра на командите, разпознава се, изчислява се дължината на командата и се изпълнява. Програмният брояч отново се актуализира, така че да сочи следващата команда. Някои от инструкциите (иапример за преход) променят съдържанието на програмння брояч. Това може да става винаги (команди за безусловен преход) или в зависимост от дадено условие (команди за условен преход). В тези случан адресът на следващия КОП няма да се намнра непосредствено след последиата изпълнена команда, а някъде другаде в паметта.

Последоввтелността от байтове в паметта, състояща се от команди за микропроцесора, се иарича *машинна програма*. Дължината на инструкциите за МП 6502 е 1, 2 или 3 байта. Писането на машинни програми е труден процес. Програмистът трябва добре да знае шестнайсетичния код на всички КОП. Той трябва да се съобразява с конкретното разположение на програмата в паметта. Така например, ако една команда за преход предава управлението (променя адреса в програмния брояч) 5 ииструкции назад, то при вмъкване на шеста команда между тези пет трябва да се промени и командата за преход, за да сочи отново първоначалния КОП. Това е здна от причините за създаване на асемблерските езнци. При тях програмистът записва КОП не като поредица от нули и единици или шестнайсетични цифри, каквото е представянето му в машината, а използва миемонични кодове. Адресите на операндите се записват като символични имена.

Написаната на Асемблер програма се траислира, т. е. превежда се на машинен език от специална програма (транслатор), наречена също асемблер.

Ето как изглежда в паметта една машинна програма *(виж фиг. 4)*:



КОП — код на операция, допустим за микропроцесора;

ДАННИ (d) — това могат да бъдат самият операнд, с който се извършва определено действие, адресът на операнда или данни, необходими за получаването на адреса на операнда.

Машинната програма трябва да започва задължително с познат на микропроцесора КОП. Това се налага от факта, че и КОП, и данните са все еднобайтови стойности, записани една след друга в паметта. Прочетен веднъж, кодът на операция носи в себе си информация за вида на операцията, за дължината на командата (КОП+ДАННИ) и за вида на адресацията. След прочитането на данните, програмният брояч се увеличава с толкова, колкото е дължината на изпълнената инструкция, или (при команди за преход) се зарежда с нова стойност. Микропроцесорът прочита байта от заведения в програмния брояч адрес. Това е следващата за изпълнение машинна команда.

Ясно е, че за изпълнението на различните команди микропроцесорът изразходва различно време. То зависи както от дължината на командата, така и от вида ѝ. Ако например чрез една инструкция сме наредили на микропроцесора да зареди акумулатора със съдържанието на клетка от паметта, за това ще са необходими няколко стъпки (такта). В първата стъпка ще бъде разпознат кодът на операцията.

В още две стъпки (ако адресът на клетката е 16-битов) ще се изчисли и ще се изведе на адресните шини адресът на клетката. Ще трябва и още една стъпка, за да се прочете и зареди в акумулатора съдържанието на клетката. Ако пък адресът на клетката е посочен неявно (чрез съответен КОП), стъпките само за формиране на адреса са повече от две. И най-кратките команди при МП 6502 изискват за изпълнението си 2 такта (машинни цикъла). Това означава, че се изпълняват за около 2 микросекунди.

НОВИ КНИГИ

МИКРОКОМПЮТЪРНА ТЕХНИКА ЗА ВСИЧКИ

В Държавно издателство „Техника“ се подготвя нова поредица, наречена „Микрокомпютърна техника за всички“. Тя е адресирана към младите читатели, но може да бъде ползувана от всички, които искат да навлязат в света на компютрите, да придобият втора, компютърна грамотност. Книгите са ориентирани предимно към персоналния компютър Правец-82, но някои могат да се ползуват и за усвояване на знания и навици за работа с произволни електроноизчислителни машини. Поредицата се издава със съдействието на Централния комитет на ДКМС и Министерство на народната просвета.

Какво ще представляват отделните издания от поредицата?

Още от първата книга — „Работа с персонален компютър“ — могат да се получат най-необходимите знания и умения за работа с персоналните компютри, а с втората — „Програмирането — и просто, и сложно“ — може да се навлезе в същността на програмирането.

В третата книга — „Бейсик — език на персоналните компютри“ — съвсем достъпно са представени всички команди и оператори от версията на езика Бейсик за българския персонален компютър Правец-82. Включени са много примерни програми.

Какво представлява микропроцесорът? Какво е предназначението му? На тези въпроси отговори следващата книга — „Микропроцесорът — сърцето на микрокомпютъра“. В нея подробно са разгледани микропроцесорите, влагани най-често в персоналните компютри — 6502, Z80 и I 8086/8088.

Чрез книгата „Как работи Правец-82“ всеки може да се запознае по-подробно с българския персонален компютър, с начина на работа, с входно-изходната организация, с блоковата схема на компютъра, с модулите за разширение на системата.

На въпроса „Що е дискова операционна система?“ е даден отговор в шестата книга от поредицата. В нея е разгледана дисковата операционна система, с която работи компютърът Правец-82. Анализирани са всички команди за работа с файлове и връзката им с езика за програмиране Бейсик.

Интересна е и книгата „50 програми за персонален компютър“. Програмите са написани на Бейсик и могат да се използуват в обучението, науката и техниката.

В осмата книга — „Паскал за персонални компютри“ — е описан алгоритмичният език Паскал така, както се употребява при написване на програми за компютъра Правец-82.

Пълните текстови, графични и звукови възможности на компютъра Правец-82 са описани в книгата „Компютърът играе, рисува и свири“.

Какви периферни устройства могат да се включват към Правец-82? Как могат да се управляват програмно? Кога и какво устройство да се предпочете? Отговор на тези въпроси се дава в десетата книга — „Периферни устройства за персонални компютри“.

Какви са българските професионални компютри, може да се научи в книгата „Професионални компютри“. Дадени са техните особености, устройството, свързването им с външни устройства, програмирането и приложението.

Направете сами микрокомпютър! Сами?! Микрокомпютър?! Да, такова е заглавието на последната книга от поредицата. В нея е представен вариант на микрокомпютър, изграден на основата на микропроцесорната фамилия СМ 600, със съответни технически и програмни възможности, различен от разпространените у нас едноплатков микрокомпютър.

Инж. ПЛАМЕН ГАМИЗОВ

В.Р. Първите две книги от поредицата бяха пуснати за продажба през януари т. г.

Продължение от бр. 1

Инж. КАМЕН КАНЕВ

ГРАФИЧНИ ВЪЗМОЖНОСТИ

ПК ПРАВЕЦ - 82

За удобство и прегледност основният алгоритъм на програмата е представен чрез блок-схема. Вижда се, че там се извиква подпрограма за посочване на точка с курсора. Входните параметри на тази подпрограма са променливите ХС и УС, които задават текущите координати на графичния курсор. Използват се и две работни променливи — DX и DY, които определят текущата стъпка за придвижване на курсора съответно по Х и У координатите. Веднага след извикването на подпрограмата курсорът се появява в точката с координатиХС, УС, след което циклично се проверява за натисиат клавиш. При иатискането на клавиш се аиализира неговият код. Ако това е един от клавишите I, J, К, М, курсорът се изтрива чрез повторно начертаване с допълнителен цвят и се появява отново, но отместен на една позиция по Х или У, в съответствие с натиснатия клавиш. При натискане иа клавиш, различен от I, J, К, М, се връщаме към главната програма. Изходните параметри са ХС и УС — новите координати на курсора, и A$ — кодът на натиснатия клавиш.

В основната програма се проверява дали е натиснат клавишът Т (точка) или О (отсечка), при което върху екрана се начертават съответните точки или отсечки. Програмата завършва работа при натискането на клавиша Е. Ако натисиатият клавиш е различен от О, Т, Е, се дава предупредителен звуков сигнал и отново се извиква подпрограмата за посочване на точка с курсора.

А сега да кодираме на Бейсик разгледаните алгоритми:

```
100  REM  УНИВЕРСАЛНА ПРОГРАМА ЗА
     РИСУВАНЕ
110  POKE 232,0: POKE 233,3: POKE
     768,1: POKE 769,0: POKE 770,
     4: POKE 771,0: POKE 772,176:
      POKE 773,236: POKE 774,79: POKE
     775,0
130  HGR : HCOLOR= 7: ROT= 0: GCALE=
     20
140 XC = 140:YC = 80
150 XN = XC:YN = YC:PM = PM + 1
160  GOSUB 390
170  IF A$ <  > "T" AND A$ <  > "
     T" THEN 210
180  HPLOT XC,YC
200  GOTO 150
210  IF A$ <  > "O" AND A$ <  > "
     O" THEN 250
220  HPLOT XN,YN TO XC,YC
240  GOTO 150
250  IF A$ <  > "E" AND A$ <  > "
     E" THEN 370
360  END
370  PRINT  CHR$ (7);
380  GOTO 160
390  REM  ПОДПРОГРАМА ЗА ПОСОЧВАН
     Е НА ТОЧКА С КУРСОРА
400 DX = 0:DY = 0
410 XC = XC + DX:YC = YC + DY
420  XDRAW 1 AT XC,YC
430 A =  PEEK ( - 16384): IF A <
     128 THEN 430
440 A$ =  CHR$ (A - 128): POKE  -
     16368,0
450  IF A$ <  > "I" AND A$ <  > "
     И" THEN 480
460  IF YC <  = 0 THEN 430
470 DX = 0:DY =  - 1: GOTO 590
480  IF A$ <  > "J" AND A$ <  > "
     Й" THEN 510
490  IF XC <  = 1 THEN 430
500 DX =  - 1:DY = 0: GOTO 590
510  IF A$ <  > "K" AND A$ <  > "
     К" THEN 540
520  IF XC >  = 279 THEN 430
530 DX = 1:DY = 0: GOTO 590
540  IF A$ <  > "M" AND A$ <  > "
     М" THEN 570
550  IF YC >  = 159 THEN 430
560 DX = 0:DY = 1: GOTO 590
570  XDRAW 1 AT XC,YC
580  RETURN
590  XDRAW 1 AT XC,YC
600  GOTO 410
```

ЗАВЕЛЕЖКА: В ред 110 с операторите POKE се подготвя необходимата за изрисуване на курсора таблица фигури, върху чийто формат няма да се спираме.

Въведете текста на основната програма и подпрограмата и подайте командата RUN. Опитайте да нарисувате различни фигури с клавишите I, J, К, М, О и Т. Надявам се, че този начин за рисуване ви допада повече от явното задаване на координатни точки, било то чрез HPLOT, или DATA. Вероятно вече сте забелязали и една неприятна особеност на новата програма — не е предвидено съхраняване на

вече нарисуваното изображение. Това обаче лесно може да се осъществи чрез командите на дисковата операционна система. Не забравяйте, че преди да използувате тези комаиди, трябва да завършите изпълнението на програмата чрез натискане на клавиша E. За съхраияване иа изображението върху дискета използувайте командата:

BSAVE РИСУНКА, A$2000, I$2000

където РИСУНКА е името на файла с нашето изображение. Този начин за съхраняване е твърде неикономичен — по 34 сектора за всяка рисунка върху дискетата.

Зареждането иа изображението от файл с име РИСУНКА се извършва с командата:

BLOAD РИСУНКА

Преди това трябва да сме включили режима прецизна графика чрез оператора HGR или POKE. При този начин за съхраняване наслагването на изображения едно върху друго се оказва невъзможно, тъй като при зареждането на новото се изтрива старото. Нека усъвършенствуваме нашата програма по такъв начин, че изображенията да се съхраняват по-икономично върху дискетата и да могат да се наслагват едно върху друго. За това е необходимо още при чертането да се съхранява точно описание иа движението на перото, осигуряващо възпроизвеждане на изображението при нужда. Удобен формат се оказва списъкът от точки, който използувахме в оператора DATA прн предишните примери. Сега обаче списъкът няма да бъде зареден предварително, а ще се попълва в процеса на рисуване, поради което най-удобно е да го разположнм в целочислен масив P%. Чрез оператора DIM P%(1,1000) запазваме място за двете координати на 1000 точки в нашия спнсък, като текущият брой значещи точки ще се съхранява в променливата PM.

Да припомнням, че отрицателната координата X означава начало за нова начупена линня, т. е. придвнжване на перото без чертане до точката със съответните положителнн координати X, У.

В последната версия на програмата за рисуване трябва да вмъкнем следните редове:

```
120  DIM P%(1,1000):PM = 0
155  PRINT PM - 1
190 P%(0,PM) =  - XC:P%(1,PM) = Y
    C
230 P%(0,PM) = XC:P%(1,PM) = YC
```

Сега вече в масива P% ще се съхранява описание на движението на перото и чрез него то ще може да се възпронзведе. Разбира се, записаното в оперативната памет изображение може да се съхрани само до зареждането на нова програма или до изключване на захранването на компютъра. Значително по-удобно е съдържанието на масива P% да се съхрани във файл на дискета. За това добавете в програмата следния фрагмент:

```
260  INPUT "ИМЕ НА ФАЙЛ:";A$
270  IF A$ = "" THEN 360
280 D$ =  CHR$ (4)
290  PRINT D$;"OPEN ";A$
300  PRINT D$;"WRITE ";A$
310  FOR I = 1 TO PM - 1
320  PRINT P%(0,I): PRINT P%(1,I)

330  NEXT I
340  PRINT 0: PRINT 0
350  PRINT D$;"CLOSE"
```

Изпълнява се при натискане на клавиша E за край на рисуването. На въпроса „ИМЕ НА ФАЙЛ:“ вие можете да отговорите по два начина:

— с натискане на клавиша RETURN, при което вашата рисунка няма да бъде запазена на дискетата;

— с избрано от вас име нв файл и след това с RETURN, за да я съхраните на дискетата. Бъдете внимателни, защото, ако вече сте използувалн файл с това име, старото му съдържаиие ще бъде заместено от новото.

С така модифицираната програма, надявам се, ще вн бъде приятно да работите и ще си създадете цила библиотека от рисунки върху дискетата. Направете различни експернменти. Каква е дължината на файла с къщичката? Значително по-малка от 34 сектора, нали? За да възпроизвеждаме върху екрана записаните на дискета изображения, ще използуваме кратката програма:

```
100  REM   ПРОГРАМА ЗА ИЗЧЕРТАВАН
     Е НА ФАЙЛ
110  HGR
120  HCOLOR= 7
130 D$ =  CHR$ (4)
140  INPUT "ИМЕ НА ФАЙЛ:";A$
145  IF A$ = "" THEN  END
150  PRINT D$;"OPEN ";A$: PRINT D
     $;"READ ";A$
160  INPUT X: INPUT Y
165  PRINT X,Y
170  IF X < 0 THEN  HPLOT  - X,Y:
     XN =  - X:YN = Y: GOTO 160
180  IF X > 0 THEN  HPLOT XN,YN TO
     X,Y:XN = X:YN = Y: GOTO 160
190  PRINT D$;"CLOSE"
200  GOTO 140
```

Тя много наподобява един от предишните примери с тази разлика, че там данните бяха зададенн в оператора DATA, а тун са във файл. Изпълнете програмата с RUN и иа въпроса „ИМЕ НА ФАЙЛ:“ отговорете с името на вече записана от вас върху дискетата рисунка. При правилно зададено име рисунката се прочита от дискетвта н се изрисува иа екраиа, след което можете да въведете име на нова рисунка. По този начин рисунките естествено се наслагват една върху друга. При въвеждаие на празио име на файл програмата завършва.

По-долу е дедеи текстът иа кратка програма, наречена „РАЗХОДКА ИЗ КАРТИННАТА ГАЛЕРИЯ“. Както лични от името й, тя ще ви показва последователно различни рисунки, които вне трябва да подготвите предварително във файлове. Имената им се задават чрез операторите DATA в края на програмата. Броят на рисуиките се задава в ред 260.

```
100  REM  РАЗХОДКА ИЗ КАРТИННАТА
     ГАЛЕРИЯ
110  HGR
120  HCOLOR= 7
130 D$ =  CHR$ (4)
140  READ N
150  FOR I = 1 TO N
160  READ A$
170  PRINT D$;"OPEN ";A$: PRINT D
     $;"READ ";A$
```

```
180  HGR
190  INPUT X: INPUT Y
200  IF X < 0 THEN  HPLOT  - X,Y:
     XN =  - X:YN = Y: GOTO 190
210  IF X > 0 THEN  HPLOT XN,YN TO
     X,Y:XN = X:YN = Y: GOTO 190
220  PRINT D$;"CLOSE"
230  FOR J = 1 TO 1000: NEXT J
240  NEXT I
250  END
260  DATA  3
270  DATA  РИСУНКА1
280  DATA  РИСУНКА2
290  DATA  РИСУНКА3
```

При равработване иа някон по-сложни рисунки често е иужно в тях да се вмъкнат елемеити, които вече са използуваии при създаването на други рисунки. Тук предлагаме помощна програма, която позволява обединяването иа няколко рисунки или елементи от рисунки в едиа.

```
100  REM  ПРОГРАМА ЗА ОБЕДИНЯВАНЕ
       НА РИСУНКИ
110 D$ =  CHR$ (4)
120  INPUT "ИМЕ НА НОВАТА РИСУНКА
     :";N$
130  PRINT D$;"OPEN ";N$
140  INPUT "ИМЕ НА РИСУНКА ЗА ОБЕ
     ДИНЯВАНЕ:";A$
150  IF A$ = "" THEN 220
160  PRINT D$;"OPEN ";A$
170  PRINT D$;"READ ";A$
180  INPUT X: INPUT Y
190  IF X <  > 0 THEN  PRINT D$;"
     WRITE ";N$: PRINT X: PRINT Y
     : GOTO 170
200  PRINT D$;"CLOSE ";A$
210  GOTO 140
220  PRINT D$;"WRITE ";N$
230  PRINT 0: PRINT 0
240  PRINT D$;"CLOSE"
```

На въпроса „ИМЕ НА НОВАТА РИСУНКА:“ посочете иовото име иа файла.След това посочете последователно имената на рисунките, които ще бъдат включени в новосъзададената. След като изчерпите списъка от имеиа, отговорете с празно име, при което програмата ще завърши.

ЗАДАЧИ ЗА УПРАЖНЕНИЕ

1. Да се усъвършенствува универсалната програма за рисуване така, че натискането на клавиша ЛАТ или натиснат клавиш RPT и един от клавишите I, J, K, M да ускорява движението на графичния курсор.

2. Да се въведат диагонални посоки на движение на курсора при натискане на клавишите U, O, ⩽, N.

3. Курсорът да се управлява с PDL(0) и PDL(1). Възможно ли е начинът на управление на курсора да се избира автоматично от програмата?

Инж. ДИМИТЪР ЕВСТАТИЕВ

УПРАВЛЯВАЩА ПРОГРАМА МОНИТОР

Продължение от бр. 1

● Създаване и изпълнение на програми на машинен език

Създаването на програми на машинен език се налага, когато изискванията за бързодействие и икономия на памет не могат да бъдат удовлетворени от езика Бейснк. Машиннният език е единственият, който компютърът разбира. Всички останали езици трябва да бъдат „превеждани“. Създаването на програми на машинен езнк е трудоемък процес, тъй като се работи с шестнайсетични числа — машинните кодове на ииструкциите и данните. Но изискванията за време на изпълнеиие и обем памет понякога се оказват определящи. Трябва веднага да се направи уговорката, че това се отнася най-често за малки подпрограмн, които ще бъдат изпълнявани чрез програми на Бейсик. Прн разработване на големи и сложни програми се използва Асемблер.

Създаването на програмн на машинен език изисква познаване на архитектурата иа компютъра, на програмния модел иа микропроцесора 6502 и иа неговите ииструкции. В програмите иа машинен език могат да бъдат извикваи всички подпрограми на мониторя. Мониторът възприема програмите на машинен езнк като подпрограмн. Ето защо те трябва да завършват с машинен код $60. Този код съответствува на инструкцията RTS (връщане от подпрограма). Прн изпълнението на тази инструкция управлението се предава от програмата на машинен език към монитора.

Ето една програма иа машинен език, която извежда буквите от A до Z:

```
*300:A9 C1 20 ED FD 1G 69 1 C9
  DB D0 F6 60    RETURN

*300.30C    RETURN

0300- A9 C1 20 ED FD 1G 69 01
030G- C9 DB D0 F6 60
```

Общият вид на командата, с която се извиква програма на машинен език за изпълнение, е: [АДРЕС] G

```
*300G
ABCDEFGHIJKLMNOPQRSTUVWXYZ
*
```

Мониторът има команда (за *реасемблиране)*, с която „се превежда" съдържанието на последователни клетки от паметта на език Асемблер. Общият вид на тази команда е: [АДРЕС] L. Командата извежда върху екрана 20 последователни реда.

```
*300L

0300-  A9 C1     LDA   #$C1
0302-  20 ED FD  JSR   $FDED
0305-  18        CLC
0306-  69 01     ADC   #$01
0308-  C9 DB     CMP   #$DB
030A-  D0 F6     BNE   $0302
030C-  60        RTS
030D-  00        BRK
030E-  00        BRK
030F-  00        BRK
0310-  00        BRK
0311-  00        BRK
0312-  00        BRK
0313-  00        BRK
0314-  00        BRK
0315-  00        BRK
0316-  00        BRK
0317-  00        BRK
0318-  00        BRK
0319-  00        BRK
```

Всеки ред съдържа адреса на първия байт на инструкцията, нейните байтове (един, два или три в зависимост от начина на адресация) и кода на операцията (мнемоничното означение на инструкцията в езика Асемблер).

● Проверка и промяна на съдържанието на регистрите на микропроцесора

Съдържанието на регистрите на микропроцесора може да се изведе върху екрана чрез комаидата:

```
* MK-E  RETURN

 A=0A  X=FF  Y=DB  P=B0  S=F8
```

Съдържанието на регистрите може да се промени с набиране на знака „:" и последователно въвеждане на новите стойности:

```
*:B0 02  RETURN

* MK-E  RETURN

 A=B0  X=02  Y=DB  P=B0  S=F8
```

● Промяна на вида на знаците, извеждани в текстов режим

С командата I знаците могат да се извеждат върху екраиа в *инверсен* вид (тъмни знаци на светъл екран) както при изпълненне на командата INVERSE в Бейсик. Премикаването към извеждане на знаците в нормалеи вид се осъществява с командата N (аналогична иа командата NORMAL в Бейсик).

● Иициализацня на слотовете

Общият вид иа командата за инициализация иа слотовете като изходи е: [НОМЕР НА СЛОТА] МК—Р. При изпълнение на командата МК—Р в клетки CSWL и CSWH ($36—$37) се записва иачалният адрес на мониторната подпрограма за извеждане на знак върху екраиа COUT1 ($ FDF0). При избор на другите слотове в тези клетки се записва адрес $ CSOO, където S е номерът на слота (от 1 до 7). Това е началиият адрес на паметта иа интерфейсната платка, поставена в съответния слот. Компютърът ще стартира програмата, записана в тази памет, всеки път, когато поиска да изведе знак върху екрана — COUT ($ FDED).

Общият вид на командата за инициализация на слотовете като входове е: [НОМЕР НА СЛОТА] МК—К. При изпълиение на командата МК—К в клетки KSWL и KSWH ($38—$39) се записва началният адрес на мониторната подпрограма за въвеждане от клавиатура KEYIN ($ FD1B). При избор иа другите слотове в тези клетки се записва адрес $ CSOO, където S е номерът на слота (1—7). Компютърът ще извиква за изпълнение програмата записана в паметта на интерфейсната платка, всеки път, когато поиска въвеждане от клавиатура — RDKEY ($ FDOC).

● Събиране и изваждане

Мониторът позволява събиране и изваждаие на еднобайтови числа. Общият вид на тези командн е:

[СТОЙНОСТ] + [СТОЙНОСТ]
[СТОЙНОСТ] — [СТОЙНОСТ]

Ето няколко примера:

```
*20+13          *4A-C
=33             =3E

*FF+4           *3-4
=03             =FF
```

СЛЕДВА

Както обещахме в миналия брой, откриваме новата постоянна рубрика с материали за Правец-8Д. С това изпълняваме желанието на повече от стотe читатели, които ни писаха. Интересът към най-младия член на фамилията Правец е разбираем, защото той ще бъде първият действително „персонален" компютър, т. е. компютър, който ще можем да купим за домашно ползване.

Първоначално ще ви запознаем с особеностите при работата с него и най-вече по какво се различава от вече познатия Правец-82, а по-късно ще увеличим броя на програмите, предназначени за него. Представяме ви и автора на поредицата — Димитър Вавов, директор на Програмната къща към Научно-производствения комбинат по микропроцесорна техника в Правец.

ДОМАШНИЯТ КОМПЮТЪР ПРАВЕЦ - 8Д

ДИМИТЪР ВАВОВ

Научно-производственият комбинат по микропроцесорна техника в Правец вече трета година усвоява производството на фамилия иерсонални компютри. Най-новото изделие на комбината е първият български домашен компютър Правец-8Д.

„Малкият" Правец, както често го наричат специалистите, е разработка на Базата за развитие и внедряване към комбината. Той е типичен представител на съвременните домашни компютри. Предлага функционално развита версия на популярния език от високо ниво Бейсик, оперативна памет за 65536 символа (64 Кбайта или около 30 страници текст), три звукови канала, осем цвята и графика с висока разделителна способност. Работи със същия микропроцесор като Правец-82 — осембитов СМ 530 или 6502.

От Правец-8Д се очаква да стане едно *икономически оправдано* средство за въвеждане на компютърна грамотност. Той позволява пълноценно обучение, практика по програмиране, навлизане в безкрайния, фантастичен свят на различните компютърни игри. С известна допълнителна периферия (печатащо и флопидисково устройство) Правец-8Д може да се използува и за водене на кореспонденция, обработка на документи, текстове и други по-професионални цели.

Какви са перспективите пред домашния компютър?

Той ще се предлага в търговската мрежа на страната срещу цена, по-ниска от 500 лв. С размери 320 х 240 х 60 мм и тегло под 1 кг Правец-8Д наистина защищава физически характеристиката „малък". Малка е и консумираната мощност — 30 вата.

Всъщност домашният Правец-8Д се предлага като компактен централен процесор с клавиатура, входно-изходни интерфейси и захранване. В търговския комплект влизат книга-документация, демонстрационна касета и кабел за връзка с телевизор. През тази година се предвижда производство на около пет хилядн броя. Това определя и мястото на Правец-8Д в плановете за масова компютризация а нашата страна.

Бурното развитие на микроелектрониката породи голямо разнообразие от съвременнии микрокомпютри. След 1980 г. се наложи и терминът *домашен компютър*, който характеризира най-малките машинн.

Домашните компютрн се намират в най-долната част на скалата за тези изделия. Типично за тях е стандартното използуване на бнтов касетофон като аъншна памет. Над домашните се намират персоналните компютри с флопндисковн устройства, персоналните компютри с унничестър-дискове н накрая се стига до разширенн конфигурации на мултимикрокомпютри.

Ако трябва само с няколко показателя да опишем домашния компютър, то сега най-важното според нас е:

- **ниска цена (до 500 лв);**
- **работа с битов телевизор,**
- **пълноценни програмни и изчислителни възможности,**
- **ограничена и неудобна външна памет ва касетофон.**

От тези показатели най-привлекателна е ниската цеиа, а най-неприятен момеит е практическата работа с касетофон. Естествено двата въпроса са пряко свързани. Подобреиата външна памет водн неизбежно до повишена цена на системата.

Както всяка масова електроника, домашният компютър има своите силни и слаби страни. Това предопределя и целите, за които може да се използува ефективно:

— *за обучение* **(в различни направления, включително по програмиране, учебни дисциплини, езикова подготовка и т. н.),**

— *за игри и забава* (възможностите за запълване на свободното време са наистина изумителни),

— *в бита* (за решаване на задачи от личен характер, за управление на различни битови уреди, например звукови уредби, отоплителни прибори и други).

Акцентът върху забавната и образователната страна на компютъра внася някои характерни изисквания. Хубави игри и ефективно обучение не стават без качествена графика, цветове и звук. Това по правило налага предвиждането на допълнителни графични и музикални команди, често реализиран и апаратно. Така например в Правец-8Д има седем допълнителни звукови команди и Бейсик, които използуват възможностите на специализиран чип — музикален синтезатор.

Ниската цена предопределя типа на външния носител на данни. В момента (а очевидно и поне още няколко години) битовият касетофон е най-евтиното и разпространено техническо средство за запаметяване на компютрирана информация. Тъй като не е специализирано за тези цели, то естествено не е най-удобното. Независимо от това единственият начин за постигане на иска цена при домашните компютри е да се разчита на вече закупения в дома касетофон и телевизор.

Производителите на домашни компютри открито се ориентират именно към потребители, които вече имат другите два задължителни за компютърната система елемента — екрана (телевизора) и външната памет (касетофона). В противен случай, ако с компютъра трябва да се купят и специален телевизор, и касетофон, цената и удовлетворението „домашен компютър“ става приблизително тройна.

А сега по-конкретно за Правец-8Д. Цифрата 8 показва разредността на микрокомпютъра (осембитов), а Д означава домашен. Кутията на Правец-8Д съдържа два основни елемента от микрокомпютърната система:

- платка с процесор, памет, език Бейсик, системно управление,
- клавиатура за въвеждане на програми и данни.

На гърба на кутията има няколко извода — това са входно-изходните интерфейси. На самата кутия има ориентиращи символи за функциите на отделните изводи. Най-левият от тях се свързва с антенния вход на телевизора. Тук може да се работи с черно-бял и PAL сигнал.

Освен с обикновен телевизор Правец-8Д може да се използува и с цветен монитор. За тази цел вдясно от извода за телевизор се намират изводът RGB. Мониторът е специализиран видеоустройство за компютри с по-добро качество на изображението и естествено с по-висока цена.

Следващият куплунг е за връзка с касетофон. Правец-8Д използува като външно запомнящо устройство обикновен битов касетофон. С него се записват и четат програми и данни, представени като кодиран звуков сигнал. Удобно е, ако касетофонът има брояч — това улеснява търсенето на определена програма.

До извода за касетофони се намира извод за връзка с печатащо устройство (принтер). Тук може да ползуваме всеки принтер, разполагащ с паралелен интерфейс тип Centronics. Необходим е само подходящ кабел. Връзката с принтер е много съществена за професионалното използуване на компютъра. Тук трябва да отбележим, че Правец-8Д използува същата кодировка на символи както Правец-82, т. е. всеки принтер с кирилица и латиница за Правец-82 може да се използува пряко и за Правец-8Д.

Последен, най-вдясно, е универсалният извод за допълнителни устройства, който дава възможност за включване на флопидисково устройство, модем, джойстик. Както вече подчертахме, флопидисковото устройство тук е изключение. Нормално за домашните компютри се използуват най-малките дискове (формат три инча), каквито засега не се произвеждат у нас. Освен това цената на флопито е съизмерима с цената на самия компютър.

Клавиатурата на Правец-8Д изглежда приблизително така и позволява работа с главни букви на кирилица и латиница.

```
I      % I &  ( )  + Ю Ч
1 2 3 4 5 6 7 8 9 0 = — —
ОСВО W E P T   I О Р Ш Щ
  Я В Е Р Т Ъ У И О П С
МК А      С Н І К І : "
  А С Д Ф Г Х И К Л ; .
 Х С V В   М , " /
З Ь Ц Ж В Н М ?
```

Традиционният за пишещите машини клавиш „горен регистър“ тук превключва от кирилица към латиница и обратно. Освен буквите, цифрите от 0 до 9 и специалните символи като , " : ! ? и т. н. на клавиатурата на Правец-8Д има още няколко допълнителни клавиша. От тях най-често използуваният е RETURN. Неговото натискане обикновено означава край на въвеждането

(съответно на режима на чакане за компютъра) и начало на обработка на въведеното в компютъра.

Друг много важен клавиш е МК. Той се използува винаги заедно с друг клавиш. МК се държи натиснат, докато се натисне и отпусне и вторият клавиш. Това е прието да се означава като „МК-С“, т. е. едновременно натискане на МК и С.

Третият специален клавиш — ОСВ, винаги се натиска самостоятелно, но се използува в комбинация с друг клавиш. В този случай първо се натиска ОСВ, след това ОСВ се отпуска, преди да се натисне другият клавиш.

Бутонът RESET се намира на долната страна на компютъра и се използува за аварийно прекъсване на програмата. Той прекратява работата на изпълняваната програма, като предава управлението на Бейсик. Програмата остава в паметта. Бутонът е разположен отдолу, за да се избегне случайното му натискане.

За да се облекчи движението на курсора в четирите посоки по екрана, върху клавиатурата на Правец-8Д има четири специализирани клавиша със стрелки. С тях може да се коригира набран текст, да се управлява движението на обект от игра и т. н. При Правец-82 няма такива клавиши.

При натискане на някой клавиш за време, по-голямо от 2,5 секунди, се включва функцията за автоматично повторение на символа. Повторението спира с отпускане на клавиша (за сравнение — при Правец-82 това става със самостоятелен клавиш RPT, натискан едновременно с друг клавиш).

Накрая трябва да се кажат две думи и за програмното осигуряване. Основно място тук заемат различните игри. Особен интерес представляват програмни езици като Лого и Форт. Възможно е използването и на отделни приложни програми, например за текстообработка, за научни експерименти и изследвания и т. н.

СЛЕДВА

Мюнхаузенова история

КАК СЕ ЗАРЕЖДА ДОС

СЛАВЧО ИВАНОВ

ПРОБЛЕМЪТ

Когато се включи захранването на Правец-82, компютърът още не умее нито да управлява дискетното устройство, нито да обработва записите върху дискетите. Тези знания той получава от дисковата операционна система — специалната програма, на която конструкторите отделят особено внимание.

А тя настина заслужава привилегированата си позиция. Тази програма не бива да се записва в постоянната памет на компютъра, защото една операционна система дава едни възможности, друга — други. А когато се решават различни задачи, винаги е разумно да се избира най-подходящата, нали?

Тогава защо различните операционни системи не се записват върху дискети?

Ви било хубаво, но компютърът още не знае да чете дискети. Изправени пред тоя проблем, програмистите го решили досущ като барон Мюнхаузен. Нали си спомняте: дето се извадил саморъчно от блатото, дърпайки се за косите, а с ботушите изтеглил и коня. Върху приблизително същата логика е изградена стратегията на ДОС: програмата да се самоизвлича на няколко етапа.

Най-напред се прочита един сектор, с негова помощ друг и така нататък, като всеки път от дискетата се извличат все по-големи блокове и се записват в оперативната памет. Затова сигурно се е наложил с толкова прозрачен намек английският термин за зареждане на ДОС:

BOOT,
от израза

To pull oneself up by one's bootstrape

което, буквално преведено, означава, човек да се изтегли сам за каишите на ботушите.

Тук ще разкажем как се зарежда ДОС 8.3 и ще опишем четирите й фази на зареждане.

ПЪРВА ФАЗА

Освен с автоматично стартиране (със самото включване на захранващото напрежение) процесът на зареждането може да започне, като от клавиатурата се подаде една от петте възможни команди:

IN#6, PR#6, C600G, 6МК-Р и 6МК-К

Първите две команди се използуват в режим Бейсик, останалите — в Монитор[1], с което управлението се предава на програмата, записана в ROM, монтиран на управляващата платка за дискетното устройство, включено в слот #6. Ако слотът е друг, шестицата в горните команди се заменя с неговия номер.

В ROM е записана машинна програма с дължина 256 байта. При изпълнението си тя принуждава рамото на четящо-записващата глава на дисковото устройство да застане над писта 0 (оттам и тракането при системите с механично позициониране на главата — от ударите на рамото в упора), да прочете сектор 0 и

[1] *Виж поредицата „МОНИТОР“ от инж. Димитър Евстатиев.*

да го запише в паметта от адрес $800 нагоре.

ВТОРА ФАЗА

След това в асемблерсиата[2] инструкция

JMP $800

управлението се предава на адрес $800, т. е. на току-що записаната програма. Втората степен от зареждането на ДОС представлява изпълнението ѝ. Тя също е дълга само 256 байта, но използувайки умело като подпрограма част от първата степени (вече записана в паметта на адрес $800), прочита едни след друг следващите 9 сектора от писта 0 на дискетата — секторите от 1 до 9.

ПЪРВА РАЗЛИКА МЕЖДУ ОСНОВНА И ПРОИЗВОДНА ДИСКЕТА

Деветте сектора се записват на едното от две възможни места в паметта. Изборът се прави в зависимост от произхода на дискетата, от която зареждаме ДОС — дали тя е производна дискета

SLAVE DISKETTE,

или е основната системна дискета

SYSTEM MASTER DISKETTE.

Тук се налага малко отклонение. Трябва да се знае, че ДОС се записва върху дискетите по два начина: инициализиране с командата

INIT

и чрез специалнзираната програма

MASTER CREATE,

с която се модифицира вече инициализираната дискета. В първия случай се получава производна дискета, във втория се възпроизвежда основната системна дискета.

Да започнем подред. Когато дискетата е производна, деветте сектора се записват на първите девет страници от паметта (по 256 байта на страница), разположени веднага след последния адрес на ДОС в компютъра, от който е записана производната дискета (с други думи адресът е променлив). Например — ако дискетата е била инициализирана чрез компютър с капацитет на паметта 32 Кбайта, деветте сектора ще бъдат записани от адрес $7700 до $8000. А ако паметта е била 16 Кбайта, това адресно поле вече става от $3700 до $4000.

Когато обаче зареждаме от основната системна дискета, деветте сектора винаги се зареждат на едно и също място — от $3700 до $4000 (като при памет 16 Кбайта).

В процеса на зареждането на втората степени още веднъж се прочита сектор 0 от писта 0 на дискетата и се записва в паметта веднага след втората фаза (в случая с основната дискета — от адрес $3600). Това е въпрос на предвидливост — ако ще се инициализира нова дискета, да има готов запис на първата степен в паметта, който да бъде прехвърлен върху писта 0 от сектор 0 (предишният запис от адрес $800 ще бъде изтрит). Когато записването на втората фаза завърши, управлението се предава върху самата ѝея, за да продължи зареждането на ДОС.

ТРЕТА ФАЗА

Където и да са записани деветте сектора, те съдържат третата фаза. Тя се изпълнява на два етапа. В първия се прехвърля главната зареждаща програма, а във втория —

RWTS (READ/WRITE/TRACK/SECTOR)

— блокът от рутинии подпрограми на ДОС, позволяващи достъп до дискетата на равиище писта/сектор. Дотук четящата глава стоеше неподвижно, защото всичките необходими сектори бяха от една и съща писта — 0. Но идва ред да се прочетат останалите секторн на ДОС от другите две писти — 1 и 2. Затова главната зареждаща програма извиква блока

RWTS.

Той позиционира рамото над писта 2, прочита от нея сектор 4 и го записва веднага под първата фаза (при осиовна системна дискета — от адрес $3500). След това втората фаза на процеса влиза в цикъл. Един след друг в низходящ ред се прочитат следващите 26 сектора

3, 2, 1 и 0 от писта 2,
от F до 0 на писта 1
и от F до A на писта 0.

Последният сектор (A от писта 0) се записва на адрес $1BOO — в случая с основна системна дискета. Тези 27 сектора съдържат програмите на осиовните дейности на ДОС и подпрограмите за работа с файловете.

По същество целият ДОС вече е прехвърлен в паметта и ако зареждането е от производна дискета, процесът завършва дотук.

ВТОРА РАЗЛИКА МЕЖДУ ОСНОВНАТА И ПРОИЗВОДНАТА ДИСКЕТА

Когато зареждаме с основната системна дискета, се извършват още няколко операции. Тя дава съществено допълнително предимство: премества целия ДОС до най-старшия достъпен адрес в PAM и предоставя на оператора освободеното място в паметта. За тази цел мастер-версията на втората фаза се обръща чрез безусловен преход към преместващата програма

RELOCATOR

с дължина 512 байта (записана автоматично при зареждането на двете последни страиици на ДОС — от адрес $1BO3). При производна дискета в тня две страници се записват само иули.

ЧЕТВЪРТА ФАЗА

Преместващата програма определя обема на вътрешната памет на компютъра и след това премества на най-старшня достъпен адрес целия ДОС. При Правец-82 ДОС се презаписва от адрес $1D00 на окончателния си адрес $9D00

След това преместващата програма модифицира (иастройва по място) машнннтва програма ДОС, за да може тя да функцнонира правилно от иовия си адрес. Накрая преместващата програма се премества сама на последните две страници на новото място.

БЕЛЕЖКА НА РЕДАКЦИЯТА:

Различията между системната и производната дискета за мнозинството читатели имат предимно познавателно значение. Причината е в това, че у нас почти всички осембитови персонални компютри (имат се предвид Правец-82 и съвместимите с него) са с памет 48 Кбайта. Затова и а системната дискета на завода а Правец ие е предвидена програмата

MASTER CREATE.

Но това съображение е валидно само за изправни компютри. Когато във вътрешната памет има повреден чип, ситуацията се променя в полза на основната системна дискета. Ако разполагаме с мастер-версия, винаги можем да заместим повредения чип с последната страница на паметта. Тогава, щом ДОС се премести, тя ще се разположи непосредствено под последната страница и компютърът ще може да се ползува до ремонта с намален обем на паметта. А ако зареждаме с производната версия, компютърът става неизползваем.

Процедурата за зареждане на ДОС завършва с подпрограмата „Студен старт". Тя инициализира ДОС, установява границата

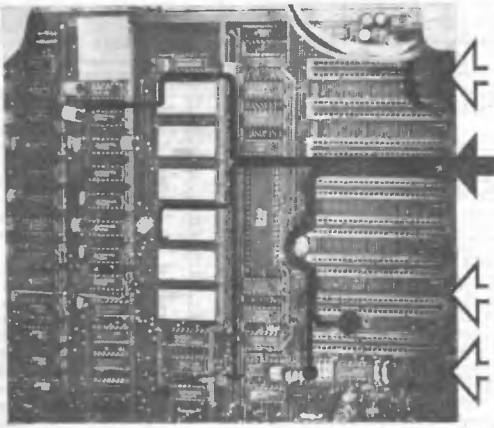
HIMEM,

изгражда векторната таблица на страница 3 от паметта и накрая стартира поздравяващата програма например

HELLO.

[2] *виж поредицата „АСЕМБЛЕР и машинен език" от Орлин Вълчев и Борислав Захариев, която започваме от този брой — на стр.14*

ПЪТЕВОДИТЕЛ ИЗ ПРАВЕЦ-82



Инж. БОРИС ВАЧКОВ
ИТКР — БАН
направление
„Персонални компютри“

БЛОКОВА СХЕМА НА МИКРОКОМПЮТЪРА

На фиг. 1 е дадена блоковата схема на ПК Правец-82.

1. Токозахранващ блок — осигурява необходимите стабилизирани постоянни напрежения (+5, +12, —5 и —12V).

2. Микропроцесор с външни буфери — микрокомпютърът използва осембитовия микропроцесор 6502. За да се повиши товароспособността на адресните и информационните му линии, те са буферирани съответно с едно- и двупосочни буфери. Това позволява включването на повече TTL товари към тях.

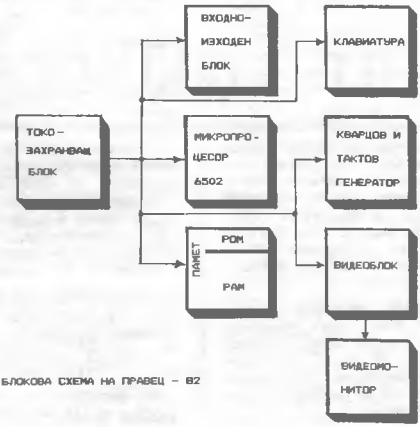
3. Памет.

— Оперативна памет (РАМ) — на основната платка е монтирана 48 Кбайта оперативна памет. Тя се състои от три реда с по 8 броя динамичеи РАМ. Микрокомпютърът може да работи и само с един или два реда от тези схеми, като при това ще има съответно 16 или 32 Кбайта оперативна памет. Това обаче значително намалява възможностите му.

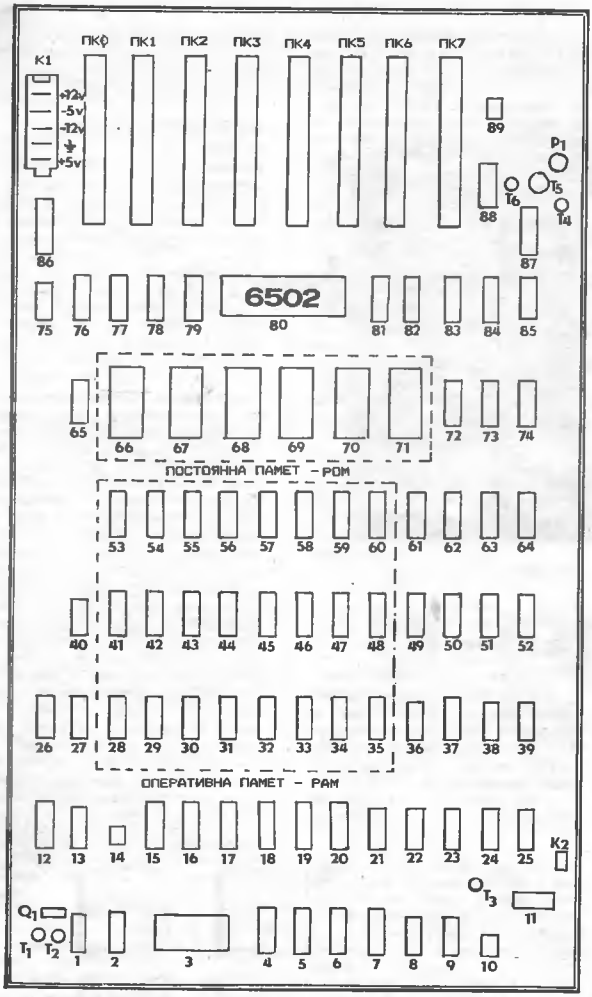
— Постоянна памет — (РОМ) — постоянната памет на микрокомпютъра е от вида ЕПРОМ. Обемът ѝ е 12 Кбайта. 2 Кбайта заема системната програма Монитор, а останалите 10 Кбайта са запълнени с интерпретатора на езика Бейсик. Тази памет се записва еднократно от завода производител.

4. Видеоблок — служи за изобразяване на информацията, намираща се на определено място в оперативната памет (видеопамет) на видеомонитора, като изработва заедно с това и необходимите сигнали за неговата работа (синхронизиращи импулси за хоризонтална и вертикална синхронизация и гасящи импулси по редове и кадри).

5. Входно-изходен блок — управлява някои входно-изходни устройства — високоговорител, касетофон, клавиатурата, входизходите за управлението на игрите, програмно-превключваемите режими на работа на видеоблока. В него влизат и монтираните на основната платка 8 входно-изходни куплунга (слота), в които се включват различни видове платки за управление на външни устройства (контролер за управление на дисково устройство, контролер за управлеине на печатащо устройство, контролер



БЛОКОВА СХЕМА НА ПРАВЕЦ — 82



за управление на цветен видеомонитор и др.), а така също и за разширяване на възможностите на микрокомпютъра (платка разширение на паметта 16 Кбайта DRAM, платка 128 Кбайта, платка с микропроцесор Z80 за работа с операционната система CP/M).

6. Кварцов генератор. Тактов генератор. Кварцовият генератор осигурява стабилизирана честота 14,25 MHZ, от която тактовият генератор изработва синхроимпулсите за работата на основните блокове.

7. Клавиатура — служи за въвеждане на буквено-цифрова информация в микрокомпютъра. Тази информация може да бъде на латиница и кирилица.

8. Видеомонитор — служи за изобразяване на буквено-цифрова и графична информация.

РАЗПОЛОЖЕНИЕ НА ЕЛЕМЕНТИТЕ ОТ ОТДЕЛНИТЕ БЛОКОВЕ ВЪРХУ ОСНОВАТА ПЛАТКА

На фиг. 2 е дадена основата платка на Правец-82 с означенията на елементите върху нея.

T1, T2 — Транзистори p-n-p 2N3906. Основни елементи на кварцовия генератор с честота 14,25 MHZ.

Q1 — Кварцов резонатор със собствена честота на трептене 14,25 MHZ. Съставен елемент на кварцовия генератор.

1 — интегрална схема (ИС) К555ЛА3 (74LS00) — четири двувходови И-НЕ логически елемента. Участват в схемите на тактовия генератор.

2 — ИС 74LS166 — осембитов преместващ регистър. Приема информацията от знаковия генератор ИС3 в паралелен и я преобразува в последователен вид.

3 — ИС EPROM К573РФ5 (2716) — използува се като знаков генератор. В него са избрани символите, съответстващи на изобразените на клавиатурата.

4 — цокъл за интегрална схема с 16 извода. Служи за свързване на лентовия кабел от клавиатурата към основната платка.

5 — ИС К555КП11(74LS257) — четири двуходови мултиплексора с три състояния. Съставна част от схемите на видеоблока за избор на графичните режими.

6 — ИС К555КП7(74LS151) — осемвходов мултиплексор. Съставна част от схемите на видеоблока.

7 — ИС К555ИР11(74LS194) — четирибитов двупосочен универсален преместващ регистър. Съставна част от схемите на видеоблока.

8 — ИС К555ТМ2(74LS74) — два D-тригера. Участват в схемите на видеоблока.

9 — ИС К555ЛЕ1(74LS02) — четири двувходови логически елемента ИЛИ-НЕ. Участват в схемите на видеоблока.

10 — ИС NE 555 — таймер. Заедно с транзистора Т3 участват в неработването на сигнала за начално нулиране при включване на токозахранването на микрокомпютъра.

11, 24, 25 — ИС К555ЛЕ1 (74LS02) — четири логически елемента ИЛИ-НЕ. Участват в схемите от видеоблока, които изработват синхроимпулсите за хоризонталия и вертикална синхронизация.

12 — ИС К531ТМ8(74S175) — четири D-тригера. Участват в схемите, изграждащи тактовия генератор на микрокомпютъра.

13 — ИС К531ЛП5(74S86) — четири двувходови логически

елемента ИЗКЛЮЧВАЩО ИЛИ. Служат като буферна схема на кварцовия генератор на микрокомпютъра и участват в схемите на тактовия генератор.

14 — ИС NE 555 — таймер. Определя честотата на мигане на маркера след оповестяващия символ]

15. 20 — ИС К555ИР11 (74LS194) — четирибитови двупосочни универсални преместващи регистри. При изобразяването на груба графика всяка участва като самостоятелен 4-битов регистър, а при фина графика двете заедно образуват 8-битов преместващ регистър.

16, 19 — ИС К555ТМ9 (74LS174) — шест D-тригера. Образуват изходните регистри на оперативната памет. В тях информацията, която ще се чете от микропроцесора или аходио-изходио устройство, се установява и е готова за обработка.

17, 18 — ИС К555КП11 (74LS257). Мултиплексори за даини. Служат за избор на даини за обработка от оперативната памет или от клавиатурата.

21 — ИС К555ТМ2(74LS74) — два D-тригера. Участват в схемите на видеоблока.

22 — ИС К555ЛИ1(74LS08) — четири двувходови логически елемента И. Участват в схемите на видеоблока.

23 — ИС К555ЛИ3(74LS11) — три тривходови логически елемента И. Участват в схемите от видеоблока, които изработват синхроиизиращите импулси за хоризонталиа и вертикална сиихронизация.

26 — ИС К555КП2(74LS153) — два четиривходови мултиплексора. Участват в схемите за адресация на оперативната памет.

27 — ИС К531ИР12(74S195) — универсален четирибитов преместващ регистър. Съставна част на тактовия генератор иа микрокомпютъра.

28 — 35,41 — 48,53 — 60 — интегрални схеми К565РУ3 (4116) — динамичен РАМ с организация 16 Кбита × 1. С тях е изградена оперативната памет на микрокомпютъра.

36 — ИС К555ЛН1(74LS04) — шест ивертора. Участват в схемите за забрана иа изходната информация от постоянната памет иа микрокомпютъра и при управлението на адресните буфери в режима DMA (директен достъп до паметта).

37 — ИС К555КП11(74LS257) — четнри двувходови мултиплексора. Участват в схемите за мултиплексиране на адресите на оперативиата памет.

38 — ИС К555ЛР11(74LS51) — участват в схемите, изработващи импулсите за хоризонтална и вертнкалиа синхроиизация.

39 — ИС К555ЛЛ1(74LS32) — четири даувходови ИЛИ схеми. Участват в изработваие иа сигнала запис-четене (R/W) иа оперативната памет.

40 — ИС К555ЛА1(74LS20) — два четиривходови И-НЕ елемента. Единият влиза в схемата на тактовия генератор, а другият участва в изработването иа сигнала ИЗБОР РАМ, който управлява мултиплексора за даннн.

49—52 — ИС К555ИЕ10 (74LS161) — двоични брояч н. Част от видеоблока.

61—63 — ИС К555КП2 (74LS153) — два четиривходови мултиплексора. Участват в адресирането иа оперативиата памет.

64 — ИС К555ИМ6(74LS283) — четирибитов двоичен суматор. Участва в адресирането на оперативиата памет.

65 — ИС К555ИД14(74LS139) — два декодера едио от четири. Участват в изработваието на стробиращите импулси за въвеждането на адресите на колоните в оперативната памет.

66—71 — ИС К573РФ5(2716) — ЕПРОМ с капацитет 2 Кбайта.

72, 73 — ИС К555ИД7 (74LS138) — едно от осем декодер/демултиплексори. Участват в декодираието иа адресите иа постоянната памет и адресите за управление на високоговорителя, касетофона, клавиатурата, входиизходите за игрите, а така също и тези иа програмно управляемите ключове за управление на видеоблока.

74 — ИС 74LS259 — осембитов адресируем регистър. Участва в схемите от вндеоблока за избора иа различните му режими на работа (първа и втора страиица режим с ииска разделителна способност; първа и втора страница графика с висока разделителна способност).

75 — ИС К555ЛИ1(74LS08) — четири двувходови И логически елемента. Участват в декодирането на адресите на постояината памет.

76 — ИС К555ИД7(74LS138) — едно от осем декодер/демултиплексор. Участва в декодираието на адресите за управление иа входно-изходните куплунги, монтираии на основната платка.

77—79 — ИС 74LS367(8T97) — еднопосочни буфери. Използват се за буфериране иа адресните линии на микропроцесора.

80 — микропроцесор 6502.

81, 82 — ИС 8T28 — двупосочни буфери. Използват се за буфериране иа информационииите лини на микропроцесора.

83 — ИС К555ИД7(74LS138) — едно от осем декодер/демултиплексор. Участва в декодираието на адресите на входио-изходните куплуиги, монтнраии иа основната платка.

84 — ИС NE558 — четворен таймер. Използва се в схемите за управление на игрите.

85 — ИС К555КП15(74LS251) — осемвходов мултиплексор. Участва в схемите за управление на вградеиите входно-изходнн устройства.

86 — ИС К555КП11(74LS257) — четири двувходови мултиплексора. Участват в схемите за управление иа оперативната памет.

87 — Цокъл за иитегралиа схема, в който се включват стиковe за управление иа игрите.

88 — ИС К555ТМ2(74LS74) — два D-тригера. Участват в схемите за управление на високоговорителя.

89 — ИС μA741 — операционен усилвател. Усилва входния сигнал от касетофон.

Т4 — Траизистор 2Т3608 (n-p-n) — служи като смесител на видеоинформацията, синхронизиращите импулси и цветовия сигнал, от колектора му се получава комплексеи видеосигиал, който се подава иа видеомонитора.

Т5 — Транзистор 2Т6551 (n-p-n) — участва в схемите за управление на високоговорителя.

Т6 — Транзистор 2Т3608 (n-p-n) — участва в схемите за управление на високоговорителя.

К1 — свързващ куплунг за подаване на стабилизираните иапрежения от токозахранващия блок към основната платка на микрокомпютъра.

К2 — свързващ куплуиг за високоговорителя.

ПКО — ПК7 — периферии куплунги с 50 извода за включване иа различни платки за управление иа външии устройства и за разширение на възможностите иа микрокомпютъра.

Р1 — потенциометър 470 ома, който служи за регулиране иа нивото на видеосигиала, подаван към видеомонитора.

РАЗШИРЯВАНЕ НА ПАМЕТТА

Инж. ПЕТЪР ПЕТРОВ
ИТКР — БАН
направление
„Персонални компютри“

I. ОСНОВНИ ПРИНЦИПИ

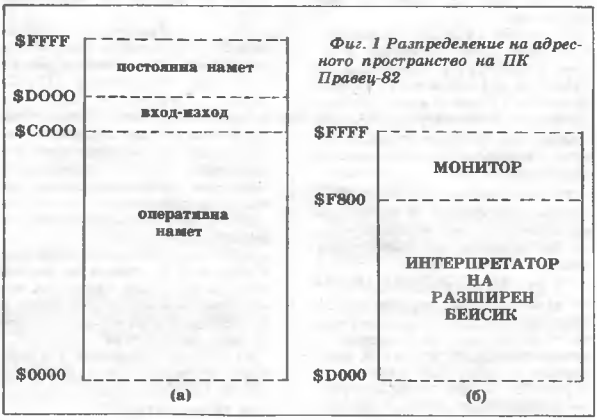
Основен проблем в архитектурата на 8-битовите персонални компютри (ПК) е проблемът за правилното разпределяне на адресното пространство на използвания микропроцесор между оперативната и постоянната памет и входно-изходните устройства. Тъй като адресното пространство, предназначено за използване от входно-изходните устройства, е много ограничено, въпросът практически се свежда до търсене на оптимално съотношение между обемите на оперативната и постоянната памет на компютъра.

Не е трудно да се направи изводът, че едва ли съществува такова съотношение между оперативната и постоянната памет, което е еднакво подходящо за всички възможни приложения на компютъра. И наистина прекомерно голямата постоянна памет, включваща резидентен език за програмиране, резидентна операционна система и дори резидентни приложни програми, използващи една и съща база данни, опростява работата с компютъра, но същевременно го превръща в тясно специализирана система. Обратно, ако съдържанието на постоянната памет се сведе само до една програма за първоначално зареждане, гъвкавостта на ПК се увеличава (създава се възможност за използване на цялото налично системно и приложно програмно осигуряване), но паралелно с това се затруднява неговото обслужване.

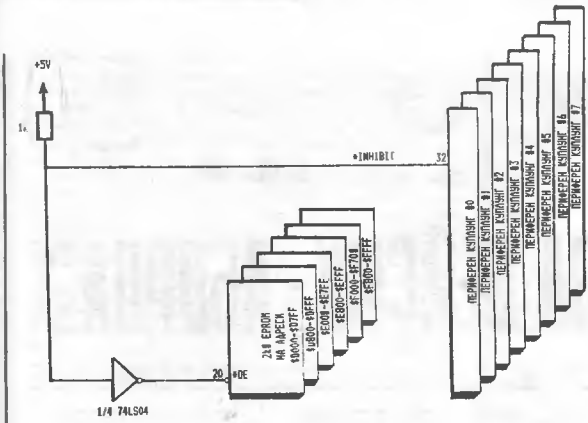
На *фиг. 1а* е показано разпределението на адресното пространство в Правец-82. Вижда се, че оперативната памет заема 48 Кбайта, входно-изходните устройства — 4 Кбайта, а постоянната памет — 12 Кбайта. На *фиг. 1б* е показано съдържанието на постоянната памет.

Като се изхожда от основните приложения на Правец-82, свързани с използването на резидент-интерпретатор на разширен Бейсик и дисковата операционна система (ДОС), може да се приеме, че това разпределение на адресното пространство на компютъра е близко до оптималното. Използването обаче на други програмни езици като Целочислен бейсик, Паскал, Лого и т. н. и на други операционни системи (като CP/M например) прави безпредметно присъствието на интерпретатор на Разширен бейсик в постоянната памет на компютъра, а следователно и на самата постоянна памет. Ето защо в конструкцията на Правец-82 е предвидена възможност за нейното изключване. За целта на 32-о перо на всички периферни куплунги е изведен сигнал *INHIBIT (звездичката пред името на сигнала означава, че неговото ниско ниво е активно).

Този сигнал се изработва от допълнителните модули и след инвертиране се подава като забрана на всички схеми на постоянната памет *(виж фиг. 2)*. Това дава възможност на допълнителните модули с надаването на логическа нула (ниско ниво) по линия * INHIBIT да блокират постоянната памет на компютъра и да използват освободеното адресно пространство. Блокирането на постоянната памет се извършва изключително под програмен контрол. Недопустимо е то да се извършва както от монитора, така и от програма, написана на



Фиг. 1 Разпределение на адресното пространство на ПК Правец-82



Фиг. 2. СИГНАЛ *INHIBIT В ПК „ПРАВЕЦ 82"

Разширен бейсик, защото в такъв случай компютърът блокира поради липса на управляваща програма.

В частност възможностите, които предлага блокирането на постоянната памет на компютъра, могат да се използуват за разширяване на неговата оперативна памет. На тази база са изградени два модула: модул 16 Кбайта РАМ, или т. нар. езикова платка, и модул 128 Кбайта РАМ. Изработването на сигнал *INHIBIT в тези случаи е свързано със следните съображения:

● На първо място е фактът, че използваните интегрални схеми, с които е изградена паметта на въпросните два модула, са с организация съответно 16 Кбайта × 1 и 64 Кбайта × 1, а освободеното адресно пространство е само 12 Кбайта. Ето защо в модул 16 Кбайта РАМ е използвана такава организация на паметта, при която в определени области от адресното пространство, на един и същ адрес, има повече от една клетка памет.

Тези области от *паралелна* памет се наричат банки, а използваната техника — *банкова организация*. Характерно за нея е това, че превключването на отделните банки се извършва под програмен контрол и че в определен момент от време е разрешен достъпът само до една банка. Погледната в по-широк план, замяната на постоянната с оперативна памет е също пример на банкова организация.

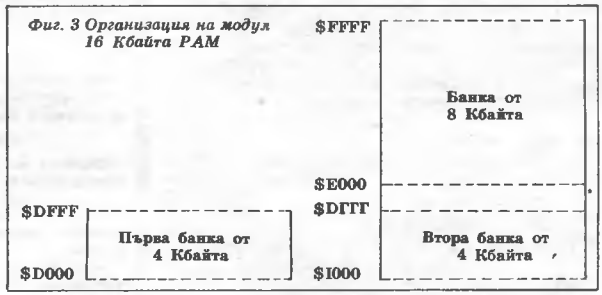
На *фиг. 3* е показано разпределението на адресното пространство на модул 16 Кбайта РАМ. То е разделено на две части. Първата е с обем от 8 Кбайта и започва от адрес $E000, а втората е с обем от 4 Кбайта, започва от адрес D000 и е дублирана. При разрешаване на достъпа до модула винаги се избира областта от 8 Кбайта. Коя от двете банки от 4 Кбайта ще се избере заедно с нея, се определя от едни *програмно-управляем ключ*.

Тази организация на модул 16 Кбайта РАМ позволява да се решат два основни проблема. Първо, заемат се освободените 12 Кбайта и се създава непрекъснато адресно пространство, което е важен фактор за стабилната работа на компютъра, и второ, ефективно се използват интегралните схеми, с които е изграден модулът.

При изграждането на модул 128 Кбайта РАМ концепцията за банкова организация на паметта е задълбочена. Модулът е съставен от 8 банки по 16 Кбайта, всяка от които е точно копие на модул 16 Кбайта РАМ. Те се превключват с 3-битов програмно управляем ключ.

● Второто важно съображение, което е взето под внимание при проектирането на двата модула, е различният характер на постоянната и оперативната памет на компютъра. Докато от първата може само да се чете, втората е достъпна както за четене, така и за запис. Това означава, че не е необходимо да се забранява постоянната памет, когато ще се извършва запис в допълнителната оперативна памет. Ето защо са създадени условия достъпът до нея да се извършва поотделно за операциите *четене* и *запис*. Взети са и мерки за предотвратяване на случаен запис в паметта на модулите, като е предвидена възможност за забрана на достъпа до тях.

СЛЕДВА



Фиг. 3 Организация на модул 16 Кбайта РАМ

ТАЙНСТВЕНИЯТ ПАРАМЕТЪР

Така е кръстен адресът 214 ($00D6) в паметта на APPLE II+. Не се наемаме да гадаем основанията на авторите на операционната система, но за нас е по-интересно, че предназначението на тази клетка е идентично с предназначението на клетката на същия адрес в Правец-82. Ако в нея запишем числото 128 ($80) чрез POKE 214,128, то всички бейсикови команди ще се възприемат като RUN, включително и командата LIST. Това е едни от начините да защитите програмата си срещу любопитни погледи. За тези пък, които обичат да поглеждат в чужди, макар и защитени програми, ще съобщим, че посочената по-горе защита се снема с POKE 214,0.

И едно предупреждение — ако държите да запазите програмата си, запишете я на дискета, преди да започнете да експериментирате. След POKE 214,128 това вече няма да бъде възможно. Също така, ако желаете да запишете вече защитената програма, сторете го, преди да сте я стартирали!

Инж. ИОРДАН ИОРДАНОВ

& КУКА ЗА БЕЙСИК · АБОРДАЖ

Как да работим с повече програми в паметта

```
5  REM   ЗАРЕЖДАЩ МОДУЛ LDAMP
10 L = 24565: POKE 1014,L -  INT
     (L / 256) * 256: POKE 1015, INT
     (L / 256): REM  ЗАРЕЖДА АДРЕ
     СА НА ПРЕХОДА ПРИ &
20  FOR I = 24565 TO 24575: READ
     A: POKE I,A: NEXT : REM    3
     АРЕЖДА МАШИННА ПРОГРАМА ОТ D
     ATA
30  DATA 169,1,133,103,169,96,133
     ,104,76,102,213
40 L = 24577: POKE L - 1,0: POKE
     103,L -  INT (L / 256) * 256
     : POKE 104, INT (L / 256): PRINT
      CHR$ (4);"MONICO": PRINT  CHR$
     (4);"LOADAMP": REM   ЗАРЕЖДА
      ПРОГРАМАТА ОТ $6000
```

фиг. 1

```
5  REM   МОДУЛ AMP
10 I =  PEEK (513):I = I - 48: REM
     ЧЕТЕ КОМАНДА ОТ БУФЕРА
20  ON I GOTO 50,70
30  GOTO 60
50  POKE  - 16304,0: POKE  - 1629
     7,0: REM   ПРЕВКЛЮЧВА ОТ ТЕК
     СТОВА СТРАНИЦА КЪМ HGR
60 L = 2049: POKE L - 1,0: POKE 1
     03,L -  INT (L / 256) * 256:
      POKE 104, INT (L / 256): END
     : REM   ВРЪЩА КЪМ НОРМАЛЕН Р
     ЕЖИМ
70  POKE  - 16304,0: POKE  - 1629
     9,0: REM   ПРЕВКЛЮЧВА ОТ ТЕК
     СТ КЪМ HGR2
80 L = 2049: POKE L - 1,0: POKE 1
     03,L -  INT (L / 256) * 256:
      POKE 104, INT (L / 256): END

90  REM  ЗА ВРЪЩАНЕ КЪМ НОРМАЛЕН
     РЕЖИМ RUN 80
```

фиг. 2

Някои случайно, други — работейки с готовн програмн на Правец-82, са забелязали, че при въвеждане на & н RETURN компютърът не подава съобщение за синтактична грешка, т. е. за него това е допустима команда. За какво служи тя н как да ни върши полезна работа?

При въвеждане от клавиатурата на & интерпретаторът се обръща към две клетки от паметта на компютъра, в които се зарежда началният адрес на програмата за обработка на командата &. Клетките са 1014 и 1015 н обикновено в тях е записан адресът $FF58.

Програмните фирмн използуват тази възможност за преходн към машиннн програмн, написанн на Асемблер за МП 6502 или на език от високо ниво (Бейсик) н после компилирани със специални програмн.

Тук се предлагат два програмни модула, с които лесно може да се реализират възможностите на & н със средствата на Бейсик — без да се премннава на машинен код.

ОПИСАНИЕ НА ПРОДУКТА

Първият модул (LDAMP) (фиг. 1) извършва следните действия:

— смеия запнсания в клетки 1014 н 1015 адрес на преход към $FF58 с адреса, откъдето ще разположнм втория модул за обработка на &-командите, намален с 11 (за конкретния пример той е $6000);

— намалението с 11 клетки осигурява място, където се запнсва кратка машинна програма, показваща на интерпретатора мястото на втората програма н прн постъпил & я стартира;

— клетки с адрес 103 н 104 показват на интерпретатора откъде е разположена програмата на Бейсик, по която той да работи;

— последното действие на първия модул е да смени адреса в тези клетки с адреса на разполагане на програмата за обработка на & н да я зареди с LOAD.

Вторият модул (AMP) (фнг. 2), който се предлага, е за демонстрация на допълнителни &-команди.

Известно е, че при премннаването в графична страница 1 илн 2 съответно с командите HGR н HGR2 съответната страница се изтрнва н гаси до черно. Ако вече нма зареденн картинн н желаем да превключнм от страницата за текст към графика, без да зачернним екрана, с командн от Бейсик не можем да се справим. Изход наистина нма — чрез POKE на определенн адреси. Неприятно е, че тези адреси трябва да се помнят или човек е принуден непрекъснато да рови из справочниците. Досадно е и самото

въвеждане на командите.

Тези функции може да се възложат на допълнителни команди &1 и &2 за превключване от страница за текст към графичните страници 1 или 2 без изтриване на изображението. Командите се реализират от модул AMP, който се зарежда от първия модул LDAMP. Как работи AMP?

Преди да се обработи от интерпретатора, всеки въведен от клавиатурата ред се записва във входен буфер, разположен от клетка 512 до клетка 766. Програмата, която обработва &-командите, трябва да се обърне към този буфер и „да прочете" какво точно да извършн.

В зависимост от прочетения код програмата трябва да направи преход към съответните участъци, където се извършва действието.

След като то е извършено, трябва да се смени адресът на програмата, по която да работи интерпретаторът, с адреса на потребителската програма (клетки 103 и 104), т. е. да се върнем в нормалния режим.

НАЧИН НА РАБОТА С ПРОГРАМИТЕ

1. Двата програмни модула трябва да са разположени на един диск. Ако потребителят желае, може да добави модула LDAMP към известяващата програма, която ДОС автоматично ще изпълни след включване на компютъра.

2. Ако LDAMP не е добавен за автоматично стартиране, той се стартира по нормалния начин (с RUN). Ако е добавен, тази точка не се изпълнява.

3. След изпълненнето на LDAMP модула интерпретаторът остава във втората програма (AMP). За да се върне към нормална работа (от адрес $800), трябва да се подаде командата RUN 80.

След като тази команда се изпълни, интерпретаторът е в състояние да работи с всяка потребителска програма, разположена от адрес $800. Модулът LDAMP може да се изтрие с NEW и да се зареди по нормален път (с LOAD или BLOAD) друга програма или пък да се започне нова програма. Командите за преход &1 и &2 могат да се подават по всяко време на работа, а зръщането от графична към текстова страница трябва да става с TEXT.

НАСОКИ ЗА ДРУГИ ПРИЛОЖЕНИЯ

Ако по някакви причини програмата AMP за обработка на &-командите трябва да се разположи от друг адрес (не от $6000), трябва да се смени значението на L от 24565 на желания адрес, намален с 11.

Броячът на цикъла за зареждане на машинния модул трябва да се променн от адреса минус 11 до адреса минус 1.

В оператора DATA числото 1 и числото 96 трябва да се сменят с числа, равни на младшия и старшия байт на нолня адрес. Ако новият адрес е $ABCE, младшият байт ще се получи по формулата:

$$12\times16+14+1,$$

а старшият — по формулата:

$$10\times16+11$$

В последния ред на LDAMP програмата стойността на L трябва да се смени с новия адрес.

Втората (AMP) програма се съставя според нуждите на потребителя — все едно че ще се стартира нормално с RUN.

Други модели компютри използуват като команди на Бейсик и оператори DOKE, CIRCLE, PLAY и др. Защо да не ги добавим и в Правец-82?

Желая на всички успешно използуване на предложените нден. А може бн колегите ще предложат н нещо по-остроумно.

Б. Р. Ние пък с удоволствие ще го публикуваме.

ХВАТКИ

К. х. н. ПЕТЪР КАРАДАКОВ

СКРИТИ РЕДОВЕ В КАТАЛОГА НА ДИСКЕТАТА

Вмъкването на надписи в каталога на една дискета води до по-голяма прегледност при представянето на съдържащата се в нея информация. Например, ако предвиждаме дискетата да съдържа програми, обединени от общ тематичен признак, или еднотипни файлове, удобно е в началото на каталога да въведем съответен пояснителен надпис — заглавие. От друга страна, ако на една дискета са записани различни програмни системи, всяка от които се състои от по няколко файла, записаната в каталога информация би придобила по-удобен и прегледен вид, ако имената на файловете от всяка система са подредени последователно, а между тях се въведат надписи — разделители. Тук накратко ще опишем един от възможните начини за вмъкване на надписи в каталога на дискета, инициализирана под управлението на ДОС 3.3 за ПК Правец-82.

Системната команда CATALOG извежда на екрана на видеомонитора по едии ред информация за всеки записан върху дискетата файл. От първа до седма позиция включително на този ред нормално се появява информация, дали файлът е защитен, за неговия тип и за броя заемани сектори. От осма позиция нататък се появява името на файла, което може да съдържа до 30 знака и трябва да започва с буква. Включването на някои контролни знаци в името има интересен ефект и може да бъде използувано за вмъкване на надписи в каталога на дискетата.

Например, ако името на един файл започва с буква, следвана от 8 знака МК—Н=CHR$ (8) и от надпис, съдържащ до 21 знака включително, командата CATALOG ще изведе този надпис от първа позиция на реда, отнасящ се за съответния файл, като споменатата по-горе системна информация изобщо няма да се появи. Обяснение: първата буква от името на файла се извежда, както е нормално, в осма позиция на реда; следващите 8 символа МК—Н връщат курсора в първа позиция, откъдето започва отпечатването на надписа; ако той е по-дълъг от

⇒

8 символа, препокрива системната информация и първата буква от името на файла. Следната кратка програма иа Бейсик илюстрира как по описаиия начии в каталога на дискета може да се вмъкне иадписът „РАБОТНА ДИСКЕТА“:

```
10 D$ =  CHR$ (4):F$ =  CHR$ (8):
     F$ = F$ + F$ + F$ + F$:F$ =
     "A" + F$ + F$
20 M$ = "РАБОТНА ДИСКЕТА" +  CHR$
     (10)
30 F$ = F$ + M$: PRINT D$"OPEN"F$
     : PRINT D$"CLOSE"F$
```

Надписът, който се вмъква в каталога, е дефиниран на ред 20. В края му е прибавен символът МК—J=CHR$(10), който след извеждането на надписа премества курсора един ред надолу. Така надписът ще се окаже разделен с един ред от следващите имеиа на файлове. Надписът заглавие трябва да се въведе в каталога на дискетата преди записването на други файлове или веднага след изтриване на първия файл от каталога.

Ако заменим ред 20 от гориата програма например с

```
20 M$ =  CHR$ (10) + "----------
     " +  CHR$ (10)
```

получаваме програма за въвеждане на надпис — разделител в каталога иа дискетата.

От текста на програмата става ясно, че въвеждането на един надпис в каталога на дискетата е свързано със създаването на един празеи текстов файл, който заема един сектор от дискетата.

Вече създаденият надпис може да се изтрие най-лесно, ако в горната програма замеиим ред 30 с

```
30  PRINT D$"DELETE"F$ + M$
```

и изпълним програмата отиово.

Клавишът RESET в клавиатурата на Правец-82 има по-специално предназначение. Той с право може да се нарече авариен и не случайно е оцветен в червено. Обикновено се използува за прекъсване на изпълнението на програма при иежелателни за оператора действия или резултати. След генерирането на сигнал от неговото натискаие се стартира поредица от машиини инструкции, вследствие на което компютърът се връща в директен режим иа Бейсик, откъдето може да се провери програмата или да се зареди и стартира друга.

За много програмисти тази поредица от инструкции представлява интерес от гледна точка на възможностите за нейната промяиа. След натискане на RESET се прочита шестнайсетичният адрес $FA62, който се съхранява в т. нар. *адресен вектор* в клетки $FFFD и $FFFC. Това е началният адрес на подпрограма, която изпълнява някои системни действия като установяване иа екрана в текстов режим, прекъсване на входно-изходни действия и други, и извършва логическата операция *изключващо или* (XOR) със съдържанието на клетка $3F3 (1011) и константа $A5 (165). Ако резултатът е еднакъв със съдържаиието на клетка $3F4 (1012), се стартира дефинираната от потреби-

В редица случан е желателно действието на бутона RESET да се възприема от компютъра като командата RUN, т. е. вместо да се налезе от програмата, тя отново да се стартира. Това може лесно да стане, ако в началото на програмата си включите малката програма, която ви предлагаме тук.

Ще поясним накратко нейното действие, без да повтаряме вече казаното в статията „СЛЕД RESET“. Промяната се постига чрез отклоненне на RESET-сектора от нормалния му адрес $E003 към адрес и област от паметта, определени от програмиста. Този адрес (в нашия случай 765) е записан в двете специални клетки на Автостарт—ром 1010 и 1011. Използувани са три клетки на буфера на клавиатурата, защото те обикновено са свободни и няма опасност от застъпване с друга машинна програма, коя-

```
10  REM  ПРОГРАМА RESET - RUN
20  POKE 1010,253: POKE 1011,2: POKE
     765,76: POKE 766,102: POKE 7
    67,213
30  CALL  - 1169
40  CALL 1002
50  REM  НАЧАЛО НА ПРОГРАМАТА
```

СЛЕД RESET

теля подпрограма, чийто начален адрес се намира в клетки $3F3 и $3F2 (1011 и 1010). В противен случай се търси и зарежда ДОС от дисково устройство 1, като че ли компютърът е току-що включен.

Първият начин за промяна на действието на RESET е смяната на съдържанието на клетките $FFFC и $FFFD. Това е възможно само при използването на DRAM (езикова платка), тъй като в противен случай това са рам адреси от управляващата програма Монитор.

При втория начин трябва:

— в клетки $3F2 (1010) и $3F3(1011) да се зареди адресът на подпрограмата, която трябва да се стартира след RESET;

— да се извърши операцията *изключващо или* (XOR) на съдържанието на $3F3 с константата $A5 и резултатът да се запази в $3F4. Това може да се осъществи със следните инструкции:

```
LDA    $3F3
EOR    #$A5
STA    $3F4
```

След включването на компютъра в $3F2 и $3F3 се зарежда адресът $E000 (наричан *студен старт* и насочен към дисковото устройство), а след първото натискане на RESET — адресът $E003 (наричан *топъл старт и насочен към интерпретатора на Бейсик).*

А ето и някои примери за изменение на поредицата от инструкции след натискането на RESET. Прибавени към програмата, написана на Бейсик, те предизвикват различни действия на червения клавиш.

```
10 POKE 1012,0
```

След натискане на RESET се зарежда ДОС от дисково устройство 1.

```
10 POKE 1010, 102 : POKE 1011,213:
CALL 64367
```

RESET стартира от първия ред програмата, написана на Бейсик.

По подобен начин действието на клавиша RESET може да се промени така, че при натискането му да се извежда каталог или програма, да се инициализира дискета и други.

АНТОАН ДИМИТРОВ

ВМЕСТО RESET → RUN

RESET → MONITOR

то би могла да се разположи от следващия, класически за малки машинни програми адрес 768 ($300). В тези три клетки е записана инструкция — JMP (безусловен преход) на адрес $D566, който е входна точка на интепретатора за командата RUN. В клетка 1012 е записан т. нар. включващ байт (Power Up Byte), чиято стойност се определя чрез логическата операция *изключващо или* със съдържанието на клетка 1011 и константата 165. Тук тази операция се извършва от машинната подпрограма на адрес $FB6F (-1169) — ред 30.

Инструкцията на ред 40 (извикване на адрес 1002) е свързана с процедура за повторно включване на ДОС при регистриране на монитор. Ако това не се направи, ДОС-командите няма да се изпълняват.

По подобен начин може да се измени действието на бутона RESET, така че след натискането му да се влезе в управляващата програма Монитор. В случая в двете клетки 1010 и 1011 е записан адресът (входна точка) на Монитор 65385 или $FF69.

```
10  REM  ПРОГРАМА RESET-MONITOR
20  POKE 1010,105: POKE 1011,255
30  CALL  - 1169: CALL 1002
40  REM  НАЧАЛО НА ПРОГРАМАТА
```

Инж. ГЕОРГИ МИРЧЕВ

Н. с. ЛЮБКА ДИМИТРОВА
ДИМИТЪР ДИМИТРОВ

РЕШАВАНЕ НА СИСТЕМА ЛИНЕЙНИ АЛГЕБРИЧНИ УРАВНЕНИЯ ПО МЕТОДА НА ОПТИМАЛНОТО ИЗКЛЮЧВАНЕ

При всички точни методи за решаване на системи линейни алгебрични уравнения е необходимо в оперативната памет да се резервира място за масив с n^2 елемента. При *метода на оптималното изключване*, който също работи само с оперативната памет, е необходимо да се резервира място само за $n^2/4$ елемента. С други думи, този метод позволява да се решават системи с почти двойно повече уравнения. А това е съществено при използването на компютри с малка оперативна памет.

АЛГОРИТЪМ

Дадена е система линейни алгебрични уравнения от n уравнения с n неизвестни:

$$\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = a_{1,n+1}\\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = a_{2,n+1}\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = a_{n,n+1}
\end{aligned}$$

Методът на оптималното изключване не изисква едновременно да се пазят в оперативната памет всички елементи на матрицата от коефициентите пред неизвестните.

Разширената матрица е:

$$\begin{pmatrix}
a_{11}a_{12}\ldots a_{1n}a_{1,n+1}\\
a_{21}a_{22}\ldots a_{2n}a_{2,n+1}\\
\ldots\ldots\ldots\ldots\\
a_{n1}a_{n2}\ldots a_{nn}a_{n,n+1}
\end{pmatrix}$$

При стъпка 1 в оперативната памет се въвежда първият ред на матрицата (2) и елементите му се делят на a_{11}.

При стъпка 2 се въвежда вторият ред, изключва се x_1 от второто уравнение, елементите на втория ред се делят на a_{22} и се изключва X_2 от първото уравнение.

При стъпка 3 се въвежда третият ред на (2), изключват се X_1 и X_2 от третото уравнение, елементите на третия ред се делят на a_{33} и се изключва X_3 от първите две уравнения.

След извършване на стъпка К системата (1) придобива вида:

$$\begin{aligned}
&x_1 + \ldots + a^{(k)}_{1,k+1}x_{k+1} + \ldots + a^{(k)}_{1n}x_n = a^{(k)}_{1,n+1}\\
&0 + x_2 + \ldots + a^{(k)}_{2,k+1}x_{k+1} + \ldots + a^{(k)}_{2n}x_n = a^{(k)}_{2,n+1}\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&0 + 0 + \ldots + x_k + a^{(k)}_{k,k+1}x_{k+1} + \ldots + a^{(k)}_{kn}x_n = a^{(k)}_{k,n+1}\\
&a_{k+1,1}x_1 + a_{k+1,2}x_2 + \ldots + a^{(k)}_{k+1,n}x_n = a_{k+1,n+1}\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{n1}x_1 + a_{n2}x_2 + \ldots\ldots + a_{nn}x_n = a_{n,n+1}
\end{aligned}$$

Вижда се, че след изпълнението на к-та стъпка следващите уравнения на изходната система (1) остават непроменени. Като се отчете това обстоятелство, в оперативната памет се въвежда не цялата матрица (2) наведнъж, а последователно — по един ред на всяка стъпка. Тогава за провеждане на (к + 1)-та стъпка са достатъчни:

$$f(к) = к(n - к + 1) + n + 1$$

работни клетки от оперативната памет. Те са нужни за елементите на матрицата

$$\begin{pmatrix} a_{1,n+1}^{(k)} & \dots & a_{1,n+1}^{(k)} \\ \dots & \dots & \dots \\ a_{n,n+1}^{(k)} & \dots & a_{k,n+1}^{(k)} \end{pmatrix}$$

н за коефициентите на (к + 1)-то уравнение. Максималната стойност на f(к) е:

$$F(k) = \frac{(n+1)\ (n+5)}{4}$$

С други думи, за решаването на една система от n линейни алгебрични уравнения с n неизвестни е достатъчно в оперативната памет да е запазено място за $(\frac{n}{2})^2$ елемента!

ОБЛАСТ НА ПРИЛОЖЕНИЕ

Да отбележим, че при разглежданата изчислителна схема е необходимо a_{11} в изходната система (1) да е отлично от нула. Още по-точно, в хода на изчисленията е необходимо всичките a_{11}, a_{22}^{2}, a_{33}^{3}... да са различни от нула.

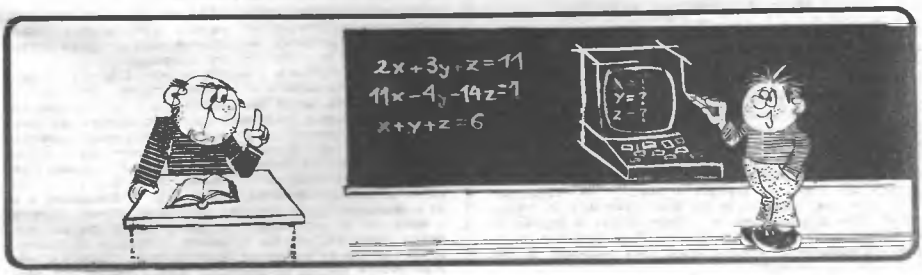
И така, стеснявайки областта на приложение, предлагаме една изчислителна схема по метода на оптималното изключване, характерна с бързина на пресмятане и икономия на оперативна памет.

В зависимост от конфигурацията на използването компютър по желание на потребителя е възможно разширената матрица (2) да бъде предварително записана по редове на външна памет (касета или дискета). Разбира се, в тези случаи трябва да се направят съответните промени в програмните редове от 140 до 180 за въвеждане на данните от външен носител.

Контролен пример:

$$\begin{aligned} 7{,}9\ X_1 + 5{,}6\ X_2 + 5{,}7\ X_3 - 7{,}2\ X_4 &= 6{,}68 \\ 8{,}5\ X_1 - 4{,}8\ X_2 + 0{,}8\ X_3 + 3{,}5\ X_4 &= 9{,}95 \\ 4{,}3\ X_1 + 4{,}2\ X_2 - 3{,}2\ X_3 + 9{,}3\ X_4 &= 8{,}6 \\ 3{,}2\ X_1 - 1{,}4\ X_2 - 8{,}9\ X_3 + 3{,}3\ X_4 &= 1 \end{aligned}$$

```
 50  HOME
 100  INPUT "БРОЙ УРАВНЕНИЯ=";N
 110 M = N + 1:K = M * M / 4
 120  DIM S(M),S1(K + M)
 130 L = 0
 140  FOR I1 = 1 TO N
 150  REM СЛЕДВА ЧЕТЕНЕ НА РЕД I1
 160  PRINT "РЕД ";I1
 170  FOR I2 = 1 TO M
 180  INPUT S(I2): NEXT I2
 190  REM ПОДПРОГРАМА ЗА РЕШАВАНЕ
      НА СИСТЕМА ЛИНЕЙНИ АЛГЕБРИЧН
      И УРАВНЕНИЯ
 200 J1 = M - L
 210  FOR J = I1 TO M
 220 R = 0:J2 = J - L
 230  IF L = 0 THEN 270
 240  FOR I = 1 TO L
 250 R = R + S(I) * S1(J2):J2 = J2
       + J1
 260  NEXT I
 270 R = S(J) - R
 280  IF J = I1 THEN R1 = 1 / R: GOTO
      300
 290 S(J) = R * R1
 300  NEXT J
 310 J3 = 1
 320  FOR I = 1 TO I1
 330 R1 = S1(J3):K1 = I1 + 1
 340  FOR J = K1 TO M
 350 J4 = J3 + J - I1:J2 = J4 - I
 360 R = S(J):S1(J2) = R
 370  IF I <  > I1 THEN S1(J2) = S
      1(J4) - R * R1
 380  NEXT J
 390 J3 = J3 + J1
 400  NEXT I
 410 L = I1
 420  NEXT I1
 430  REM КРАЙ НА ПОДПРОГРАМАТА
 440  PRINT "РЕШЕНИЕ"
 450  FOR K = 1 TO N
 460  PRINT S1(K)
 470  NEXT K
 480  END
```



АНТОАН ХЛЕБАРОВ
АТАНАС ПАНАЙОТОВ

СТАТИСТИЧЕСКА ОБРАБОТКА НА ДАННИ

ПРОВЕРКА НА СТАТИСТИЧЕСКИ ХИПОТЕЗИ ЧРЕЗ t-ТЕСТ

В социално-икономическата действителност често се налага сравняването на статистически характеристики, изчислени въз основа на извадки от две различни съвкупности от статистически единици. Регистрирани са например физически признаци (ръст, тегло и т. н.) на две групи младежи — едната от момчета, другата от момичета. Установени са различия в изчислените средна аритметична и стандартно отклонение. Чрез проверка на статистически хипотези е възможно да се установи дали тези различия са закономерни, или се дължат на случайност при подбора на единиците в извадките.

Проблемът за определяне на значимостта и наличието на закономерност в различията между статистическите характеристики на две групи статистически единици възниква в най-различни области — при сравняване на параметрите на продукцията в зависимост от използуваната технология или оборудване, при сравняване на две социални групи по даден признак и т. н.

За да се пояснят възможностите за широко практическо приложение на t-теста, ще разгледаме конкретни примери.

Нека при изследване на физическото състояние на учениците са претеглени 19 ученици — 10 момчета и 9 момичета. Необходимо е да се установи дали разликата между изчислените средна аритметична и стандартно отклонение за теглото на момичетата и момчетата е закономерна. С други думи, дали се проявява въобще при момчетата и момичетата, или се дължи на случайност, т. е. съществува само при подбраните в извадката ученици.

В предлагания варнайт на програмата се работи с предварително изчислени средна аритметичиа, стандартно отклонение и зададен обем на извадката. За посочения пример това са:

	За момчета	За момичета
1. Средна аритметична	69,5	64,22
2. Стандартно отклонение	2,99	3,03
3. Обем на извадката	10	9

Могат да се формират две хипотези:

H_0: различията между характеристикнте се дължат на случайности;

H_1: различията са закономерни и статистически значими.

Като резултат от изпълнението на програмата се получава емпиричната стойност на t и т. нар. степени на свобода.

Степените на свобода представляват броят на числата (стойностите на даден признак за всяка от статистическите единици, формиращи изследвания статистически ред), които могат да варират в рамките на определена обща сума на стойностите на реда. Ако има 10 числа, чиято сума е определена, и едното от тях е фиксирано от спецификата на метода, другите 9 са свободни да варират в рамките на общата сума. Това означава, че има 9 степени на свобода.

Чрез използване на статистически таблици (които могат да бъдат намерени във всеки учебник и справочник по статистика) се определя теоретичната стойност на t и чрез сравняването на емпиричната и теоретичната стойност се взема решение, коя хипотеза ще се приеме и какъв извод може да се направи относно сравняването на физическото състояние на момчетата и момичетата по признака „тегло“.

H_0 не може да се отхвърли, когато емпиричната стойност на t е по-малка от теоретичната (таблична). Табличната стойност се определя от координатите: брой на степените на свобода и възприет риск за грешки от провежданця изследването.

H_0 се отхвърля и се възприема H_1, когато емпиричната стойност на t е по-голяма от теоретичната.

Необходимо е да се отбележи, че отхвърлянето или приемането на дадена хипотеза за вземане на решение винаги се основава на дадена вероятност.

Тази програма може особено успешно да се използва при анализа на данни, от които предварително са изчислени елементарните статистически характеристики за някакви други цели и са на разположение при провеждането на това проучване. Необходимите входни данни са: средната аритметична и стандартното отклонение, изчислени от извадката за всяка съвкупност от статистически единици, както и обемът на двете извадки (виж бр. 1/86 „Компютър за вас“).

При вземане на решение, коя хипотеза ще бъде възприета като основа за съответните изводи, е важно да се определи как изчисляваната емпирична стойност на t-характеристиката се влияе от входните данни. В сила са следните основни зависимости:

1. При увеличаване на стандартното отклонение и при постоянни стойности на средната и обема на извадката емпиричиата стойност на t-характеристиката намалява.

2. При увеличаване на обема на извадката и при постоянни стойности на средната и стандартиото отклонение емпиричната стойност на t-характеристиката се увеличава. Това се дължи на факта, че когато увеличаваме обема на извадката, изчислената средна става по-надеждна (намалява се влиянието на случайността) и с по-голяма сигурност може да се твърди, че разликата между средните на двете съвкупности действително съществува.

Чрез предлаганата програма се сравняват успешно статистически характеристики (средна и стандартно отклонение) за две отделни, независими една от друга групи (съвкупности) от статистически единици.

В повечето случаи в реалната действителност не е възможно или най-малкото по е целесъобразно сравняваните характеристики, които могат да служат за формирането на важни изводи и заключения, да се изчисляват на основата на информация за всички съществуващи единици. В посочения пример изводът за наличието на разлика а средното тегло на момчетата и момичетата в дадена училища възраст се прави на основата на информация само за 19 младежи (единици), тъй като не е оправдано, а и не е възможно, да се измерят теглата на всички момчета и момичета на тази възраст в страната. Тук веднага възниква въпросът, доколко средната, изчислена на основата на извадка от определен брой единици, съвпада с действителната средна за всичките съществуващи единици.

В практиката този въпрос често е от голямо значение. Например доколко установената средна за даден параметър на произвеждано наделие, установена чрез намерване на 5—10% от всички произведени изделия, е действително средна за всички. Отговорът на поставения въпрос е свързан с вземането на важни решения, свързанн с управлението на качеството.

Втората част на програмата служи именно да се установи доколко средната, определена въз основа на извадката, е действително различна от средната на цялата съвкупност, която е предварително известна (например определена е за минал период или за други цели и на сегашния етап я сравняваме с действителното състояние).

Нека знаем, че необходимото средно тегло на строителни сегменти е 69,5 кг, като се допуска отклонение ±5 кг. За да се приемат изработените сегменти, комисия измерва 12 от тях, при което са получени следните действителни стойности: 68, 66, 67, 73, 69, 72, 72, 70, 69, 69, 74 и 75.

Формират се две хипотези:

H_0: средната от извадката е равна на средната за цялата съвкупност от сегменти.

H_1: средната, изчислена от извадката, е различна от тази на цялата съвкупност, т. е. налице е цилостно отклонение на действителното тегло от нормативното.

Средната, изчислена по данните от извадката, е 70,33. Стандартното отклонение е 2,64. Емпиричната стойност на t-характеристиката, изчислена чрез програмата, е 1,016 с 11 степени на свобода. Чрез използване на статистически таблици при координати — вероятност за грешка 5% и степени на свобода 11, се определя теоретичната стойност t=1,799. Тъй като t-емпирично е по-малко от t-теоретично, се налага изводът, че може да се възприема хипотезата H_0, т. е. средната аритметична, изчислена на основа на извадката, не се отклонява закономерно (статистически значимо) от средната за цялата съвкупност от сегменти (в случая предварително определена като нормативна и гарантирана от производителя).

При ползуването на програмата трябва да се има предвид, че всяка част е логически обвързана с предходните, и те трябва да се изпълняват съвместно. В последната част се дава възможност за избор от потребителя да работи с предварително изчислени характернстики или да въвежда входните данни за тяхното определяне.

```
10  HOME : VTAB 3: HTAB 5: PRINT
     "КОМПЮТЪР"; SPC( 11);"СТАТИ"
    ;"СТИЧЕСКА": PRINT  8PC( 4);
    "ЗА ВАС  ";: HTAB 24: PRINT
    "ПРОГРАМА N.2": VTAB 13
20  HTAB 3: PRINT "ИЗПОЛЗУВАНЕ Н"
    "А Т-ТЕСТОВЕ ЗА ПРОВЕРКА": PRINT
    : HTAB 8: PRINT "НА СТАТИСТ"
    "ИЧЕСКИ ХИПОТЕЗИ"
30  FOR I = 1 TO 40:E$ = E$ + " "
    : NEXT : VTAB 1: INVERSE : PRINT
    E$;: FOR I = 1 TO 20: PRINT
    " ";: HTAB 40: PRINT " ";: NEXT
    : PRINT E$: NORMAL : GET T$
40  HOME : VTAB 5: INVERSE : PRINT
    " 1 ";: NORMAL : PRINT " Т-Т
    ЕСТ ЗА ПРОВЕРК";"А ЗНАЧИМОСТ
    ТА НА ": HTAB 5: PRINT "РАЗЛ
    ИКАТА МЕЖДУ СТ";"АТИСТИЧЕСКИ
    ТЕ": HTAB 5: PRINT "ХАРАКТЕР
    ИСТИКИ НА ДВЕ НЕЗА";"ВИСИМИ
      ": HTAB 5: PRINT "ИЗВАДКИ"
    : PRINT : PRINT
```

```
50  INVERSE : PRINT " 2 ";: NORMAL
     : PRINT " Т-ТЕСТ ЗА ПРОВЕРК"
    ;"А ЗНАЧИМОСТТА НА ": HTAB 5
    : PRINT "РАЗЛИКАТА МЕЖДУ СР"
    ;"ЕДНАТА АРИТМЕТИЧНА";: HTAB
    5: PRINT "ЗА ИЗВАДКАТА И ЗА
    ЦЯЛАТА СТАТИСТИ-": HTAB 5: PRINT
    "ЧЕСКА СЪВКУПНОСТ"
60  PRINT : INVERSE : PRINT " 3 "
    ;: NORMAL : PRINT " КРАЙ"
70  VTAB 23: PRINT "ИЗБЕРЕТЕ:";: GET
    T$:ZZ% =  VAL (T$): IF ZZ% <
    1 OR ZZ% > 3 THEN : HTAB 1: CALL
     - 958: GOTO 70
80  HOME : ON ZZ% GOSUB 1000,2000
    ,3000: CLEAR : GOTO 40
1000  REM  ПЪРВИ ТЕСТ
1010 S1 = 0:S2 = 0:Q1 = 0:Q2 = 0
1020  PRINT "ЗА ВСЯКА ГРУПА ВЪВ";
    "ЕДЕТЕ": PRINT "СЛЕДНИТЕ ХА"
    ;"РАКТЕРИСТИКИ : ": PRINT : PRINT

1030  PRINT " ГРУПА 1     ": PRINT
    "СРЕДНА АРИТМЕТИЧНА     = ";:
     INPUT "";X1
1040  PRINT "СТАНДАРТНО ОТКЛОНЕ";
    "НИЕ = ";: INPUT "";S1
1050  PRINT "РАЗМЕР НА ИЗВАДКАТ";
    "А"; SPC( 3);"= ";: INPUT ""
    ;N1: PRINT : PRINT
1060  PRINT " ГРУПА 2     ": PRINT
    "СРЕДНА АРИТМЕТИЧНА     = ";:
     INPUT "";X2
1070  PRINT "СТАНДАРТНО ОТКЛОНЕ";
    "НИЕ = ";: INPUT "";S2
1080  PRINT "РАЗМЕР НА ИЗВАДКАТ";
    "А"; SPC( 3);"= ";: INPUT ""
    ;N2: PRINT : PRINT
1090 T = (X1 - X2) /  SQR ((N1 -
    1) * S1 ↑ 2 + (N2 - 1) * 82 ↑
    2)
1100 T1 = T *  SQR ((N1 * N2) * (
    N1 + N2 - 2) / (N1 + N2))
1110  PRINT "T = ";T1: PRINT N1 +
    N2 - 2;" СТЕПЕНИ НА СВОБОДА"
    : VTAB 23: PRINT "НАТИСНЕТ";
    "Е НЯКОЙ КЛАВИШ !";: GET T$:
     RETURN
2000  REM  ВТОРИ ТЕСТ
2010  DIM X(200)
2020 XO = 201
2030  PRINT "ВЪВЕДЕТЕ ДАННИТЕ:": PRINT

2040  PRINT "СРЕДНА АРИТМЕТИЧНА "
    : PRINT "ЗА ЦЯЛАТА"; SPC( 1)
    ;"СЪВКУПНОСТ = ";: INPUT "";
    M
2050  PRINT : PRINT "ИЗЧИСЛЕНИ ";
    "ЛИ СА СРЕДНА АРИТМЕТИЧНА ":
     PRINT "И СТАНДАРТНО ОТКЛО";
    "НЕНИЕ ЗА ИЗВАДКАТА ? ": PRINT
    "(ДА/НЕ) ";: GET T$: PRINT :
     PRINT
2060 S = 0:Q = 0: IF T$ = "Д" THEN
    2150
2070  GOSUB 2180
2080 T =  ABS ((M1 - M) /  SQR (V
     / N))
```

Инж. ПАВЕЛ КИРОВ

```
2090  HOME
2100  PRINT "СРЕДНА АРИТМЕТИЧНА":
      PRINT "ЗА ИЗВАДКАТА       ="
     ;M1: PRINT
2110  PRINT "СТАНД.ОТКЛОНЕНИЕ  ":
      PRINT "ЗА ИЗВАДКАТА       ="
     ; SQR (V): PRINT
2120  PRINT "ВАРИАЦИЯ           ="
     ;V: PRINT : PRINT : PRINT
2130  PRINT "T = ";T: PRINT N - 1
     ;" СТЕПЕНИ НА СВОБОДА "
2140  VTAB 23: PRINT "НАТИСНЕТЕ";
     " НЯКОЯ КЛАВИШ !";: GET T$: RETURN
2150  PRINT "СРЕДНА АРИТМЕТИЧНА":
      PRINT "ЗА ИЗВАДКАТА       ="
     ;: INPUT "";M1: PRINT
2160  PRINT "СТАНД.ОТКЛОНЕНИЕ  ":
      PRINT "ЗА ИЗВАДКАТА       ="
     ;: INPUT "";V2:V = V2 * V2: PRINT

2170  PRINT "РАЗМЕР НА          ":
      PRINT "НА ИЗВАДКАТА       ="
     ;: INPUT "";N: PRINT : PRINT
     : GOTO 20B0
21B0  PRINT
2190  HOME : PRINT "  АКО СТЕ ДО"
     ;"ПУСНАЛИ ГРЕШКА ПРИ ВЪВЕЖД"
     ;"АНЕНА ДАННИТЕ,НАТИСНЕТЕ /"
     ;" ПРИ СЛЕДВАЩОТО  ": PRINT
     "ЧИСЛО !": PRINT
2200  PRINT "РАЗМЕР НА  ": PRINT
     "ИЗВАДКАТА = ";: INPUT "";N
2210  IF N < 1 OR N >  = XO THEN
      PRINT "РАЗМЕРЪТ ТРЯБВА ДА";
     " Е ПО-ГОЛЯМ ОТ O И ПО-МАЛ";
     "ЪК ОТ ";XO: GOTO 2200
2220  PRINT : PRINT " ВЪВЕДЕТЕ ";
     "ДАННИТЕ ОТ ИЗВАДКАТА ."
2230  PRINT : FOR I = 1 TO N
2240  PRINT "СТОЙНОСТ N.";I;: INPUT
     " ";X$: IF X$ = "/" THEN 230
     O
2250  IF  ASC (X$) < 4B OR  ASC (
     X$) > 57 THEN 2240
2260 X1 =  VAL (X$)
2270 S = S +  + X1:Q = Q + X1 ↑ 2
     :X(I) = X1
22B0  NEXT
2290 M1 = S / N:V = (Q - S * S /
     N) / (N - 1): RETURN
2300  PRINT "КОЯ Е ПОГРЕШНО ВЪВ";
     "ЕДЕНАТА СТОЙНОСТ ? ";: INPUT
     "";N1: IF N1 > I THEN  PRINT
     "НЕ Е ВЪВЕДЕНА ОЩЕ !": GOTO
     2300
2310 O2 = X(N1): PRINT "ВЪВЕДЕНА"
     ;"ТА СТОЙНОСТ Е ";O2
2320  PRINT "НОВА СТОЙНОСТ ";: INPUT
     "";O1
2330 S = S - O2 + O1:Q = Q - O2 ↑
      2 + O1 ↑ 2:X(N1) = O1
2340  PRINT "ПРОДЪЛЖЕТЕ.": GOTO 2
     240
3000  HOME : END
```

```
        МАНИПУЛАТОР НА ФАЙЛОВЕ
1 - ПЕЧАТ НА ОТДЕЛЕН СЕКТОР
2 - ПОКАЗВ. НА СЕКТОР - ЕКРАН
3 - ПРОМЯНА В ОТДЕЛЕН СЕКТОР
4 - ПЕЧАТ НА ФАЙЛ
5 - ПОКАЗВ. НА ФАЙЛ - ЕКРАН
6 - ПРОМЯНА ВЪВ ФАЙЛ
7 - КРАЙ
```

МАНИПУЛАТОР НА ФАЙЛОВЕ

● ПРЕДНАЗНАЧЕНИЕ

Програмата „Манипулатор на файлове“ изисква ПК Правец-82 с едно флопидисково устройство. Наличието на печатащо устройство разрешава ползуването и на фуикциите за печат. Програмата позволява да се покаже на екрана или да се отпечата съдържанието на определен сектор или на цял файл (сектор по сектор) на ниво отделен байт. Може да се борави с всякакъв вид информация, която се показва а шестнайсетичен и текстови вид. Благодарение на това програмата може да се ползува при многобройните случаи на справки и промени в съдържанието на дискети и особено при откриване и преработване на текстове а непознати файлове.

Текстът на програмата (при своята подпрограмна реализация) с коментарите към използуваните променливи и отсъствието на сложни програмии похвати би могъл да бъде илюстрация за организацията на каталога и разполагането на файловете от ДОС 3.3.

● НАЧИН НА ИЗПОЛЗВАНЕ

След задействуване на програмата на екрана се появява менюто (фиг. 1), което дава комбинации от възможностите за работа с определен сектор или с цял файл, от едиа страна, и от друга — да се избере показването на информацията иа екран, на печат или режим иа промеии. Дискетата, с която ще се работи, трябва да се постави в устройство 1.

Печатът и показването иа екран (фиг. 2) се отличават само по броя на байтовете иа едии ред — съответно 32 и 16. Най-отляво е десетичният пореден номер в сектора на първия от групата байтове. Следват два реда. В долния всеки байт е представен с по две шестнайсетични цифри. Над лявата има точка, ако стойността на байта е по-голяма от 127, а над дисната е изписаиа буква или специален знак, ако байтът има такава стойност. В началото за всеки сектор са дадени с десетичии числа номерата на

```
 ПИСТА=17
СЕКТОР=15
П=17   С=15   ПОР.СЕКТ.ВЪВ ФАЙЛА=0

0                                          .H.E.L.L.O. . . . . .
      00110E0000000000000000070D02CBC5CCCCCFA0A0A0A0A0A0A0A0A0A0A0A0

32     . . . . . .  . . . . . .  . .M.O.D.I.F.Y.E.D. .K.E.Y.B.O.A
      A0A0A0A0A0A0A0A0A0A0A0A0A0A0A006001D04B2CDCFC4C9C6D9C5C4A0CBC5D9C2CFC1

64     .R.D. . . . . .  . .H.I.M. . . . . .
      D2C4A0A0A0A0A0A0A0A0A0A0A0A0A0A0A00500160FB2EEE9EDA0A0A0A0A0A0A0A0A0A0

96     . . . . . .  . .S.Y.S.L.I.N. . .
      A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A012001E0FB2D3D9D3CCC9CEA0A0A0

128    . . . . . .  . .B.O.W.L.I.N
      A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A001B0FB2C2CFD7CCC9CE

160    .G.*.*.*. . . . . .  . .C.Л..
      C7AAAAAAAA0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A00190FB2F3ECAE

192    .Ч.А.С.*. . . . . .  .
      DEE1F3AAA0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A010001A0FB2

224    .К.О.М.П.Р... .Н.А. .Г.Р.А.Ф.И.К.И. . . . . .
      EBEFEDF0F2AEA0EEE1A0E7F2E1E6E9EBE9A0A0A0A0A0A0A0A0A0A0A0A0A0A00A00
```

пистата и сектора и поредният номер във файла.

В режим на промени в гориата част иа екрана са показани клавишите н съответните функции. Изписано е и съдържанието на иулевия байт в сектора. Всеки байт се представя на отделен ред с поредеи иомер, шестнайсетична стойност, десетична стойност и знак. Има възможност да се върви иапред и назад а сектора и да се въведе нова стойност иа текущия байт, като предварителио се съобщи със съответния клавиш дали ще се въвеждат две шестнайсетични цифри, десетично число, следвано от RETURN, или знак. Измененията в текущия сектор могат да завършат със или без запис на променения сектор обратно на дискетата.

```
  10  REM ПРОГРАМА МАНИПУЛАТОР
  20  HIMEM: 36607: GOSUB 110: GOSUB
      1010
  30  GOSUB 2410
  40  ON RA GOSUB 2510,2515,2610,27
      10,2715,2810,60
  50  GOTO 30
  60  HIMEM: 38400: END
  99 :
 100  REM АДРЕСНИ КОНСТАНТИ
 110 AA = 37632: REM ТАБЛИЦИ
 120 AB = 37888: REM В/И БУФЕР
 130 AP = 37646: REM   ПАРАМЕТЪР-П
     ИСТА
 140 AS = 37647: REM ПАРАМЕТЪР-СЕК
     ТОР
 150 AO = 37654: REM ПАРАМЕТЪР-ОПЕ
     РАЦИЯ
 160 AF = AB + 14: REM НАЧАЛО ИМЕ
     НА ФАЙЛ 1
 170 AD = 36608: REM ТАБЛИЦА ДВОЙК
     И
 199 :
 200  REM БРОЯЧИ И ПОКАЗАЛЦИ
 210 BN = 0: REM ПОРЕДЕН СЕКТ. ВЪВ
     ФАЙЛА
 220 BB = 0: REM ПОРЕДЕН БАЙТ В БЛ
     ОКА
 250 BP = 17: REM ТЕКУША ПИСТА
 260 BS = 15: REM ТЕКУЩ СЕКТОР
 270 BD = AD: REM ПОКАЗАЛЕЦ ТАБЛИЦ
     А ДВОЙКИ
 280 BI = 32: REM  СТЪПКА
 290 BJ = 31
 299 :
 300  REM РАБОТНИ ПРОМЕНЛИВИ
 310 RS$ = "01234567B9ABCDEF"
 320 RB = 0: REM ПРОЧЕТЕН БАЙТ
 330 RC$ = " ": REM " " ИЛИ "."
 340 RF$ = " ": REM ИМЕ НА ТЪРСЕНИ
     Я ФАЙЛ
 350 RG$ = "": REM РАБОТЕН НИЗ
```

```
360 RA = 0: REM ЗА ИЗБОР НА ВЪЗМО
    ЖНОСТИ
370 RA$ = "": REM ЗА ВЪВЕЖДАНЕ НА
    СИМВОЛ
380 RH$ = "": REM РАБОТЕН НИЗ
399 :
400  REM ФЛАГОВЕ
410 FI = 0: REM НАМЕРЕНО ИМЕ НА Ф
    АЙЛ(FI=1)
420 D$ =  CHR$ (13) +  CHR$ (4)
500  RETURN
999 :
1000  REM  ТАБЛИЦИ ЗА RWTB
1010  FOR I = AA TO AA + 35: READ
    J: POKE I,J: NEXT
1020  DATA 169,147,160,10,32,217,
    3,96,0,0
1030  DATA 1,96,1,0,17,15,32,147,
    0,148,0,0,1,0,0,96,1,0,0,0,0
    ,0
1040  DATA 0,1,239,216
1050  RETURN
1099 :
1100  REM ЧЕТЕНЕ СЕКТОР ОТ ДИСКА
1110  POKE AP,BP: POKE AS,BS: POKE
    AD,1
1120  CALL AA
1130  RETURN
1199 :
1200  REM ЗАПИС НА СЕКТОР НА ДИСК
    А
1210  POKE AP,BP: POKE AB,BS: POKE
    AD,2
1220  CALL AA
1230  RETURN
1299 :
1300  REM БАЙТ В 16-ТИЧНИ ШИФРИ
1310 R1 =  INT (RB / 16):R2 = RB -
    R1 * 16
1320  PRINT  MID$ (RS$,R1 + 1,1);
     MID$ (RS$,R2 + 1,1);
1330  RETURN
1399 :
1400  REM РАЗПЕЧАТКА НА СЕКТОР
1410  PRINT "П=";BP;"  С=";BB;"
    ПОР.СЕКТ.ВЪВ ФАЙЛА=";BN: PRINT

1420  FOR I = 0 TO 255 STEP BI
1430  PRINT I; SPC( 6 -  LEN ( STR$
    (I)));
1440  FOR J = 0 TO BJ
1450 RB =  PEEK (AB + I + J):RC$ =
    " ": IF RB > 127 THEN RC$ =
    ".":RB = RB - 128
1460  IF RB < 32 THEN RB = 32
1470  PRINT RC$; CHR$ (RB);
1480  NEXT : PRINT
1490  PRINT "       ";
1500  FOR J = 0 TO BJ
1510 RB =  PEEK (AB + I + J): GOSUB
    1310
1520  NEXT : PRINT : PRINT
1530  NEXT : PRINT
1540  RETURN
1599 :
1600  REM ИМЕ НА ФАЙЛ В КАТАЛОГА
1610 BP = 17:BS = 15:RH$ = RF$: FOR
    I = 1 TO 30 -  LEN (RF$):RH$
     = RH$ + " ": NEXT
1620  FOR I = 15 TO 1 STEP  - 1: GOSUB
    1110
1630  FOR J = AF TO AF + 244 STEP
    35
1640 FI = 0:RG$ = "": IF  PEEK (J
     - 3) = 0 AND  PEEK (J - 2) =
    0 THEN  PRINT : PRINT "***НЯ
    МА ТАКЪВ ФАЙЛ": RETURN
1650  FOR K = 0 TO 29:RB =  PEEK
    (J + K): IF RB > 127 THEN RB
     = RB - 128
1660 RG$ = RG$ +  CHR$ (RB)
1670  NEXT
1680  IF RG$ = RH$ THEN FI = 1:BP
     =  PEEK (J - 3):BS =  PEEK
    (J - 2): RETURN
1690  NEXT :BP =  PEEK (AB + 1):B
    S =  PEEK (AB + 2)
1700  IF BP = 0 AND BS = 0 THEN  PRINT
    : PRINT "*** НЯМА ТАКЪВ ФАЙЛ
    ": RETURN
1710  NEXT
1720  RETURN
1799 :
1800  REM СПИСЪК ПИСТА-СЕКТОР НА
    ФАЙЛА
1810 BD = AD + 1:K = 0
1820  FOR I = 1 TO 4: GOSUB 1110
1830 AF = AB + 12
1840  FOR J = 0 TO 242 STEP 2
1850 K = K + 1: POKE BD, PEEK (AF
    ): POKE BD + 1, PEEK (AF + 1
    ): IF  PEEK (BD) = 0 AND  PEEK
    (BD + 1) = 0 THEN  POKE AD,K
     - 1: RETURN
1860 AF = AF + 2:BD = BD + 2
1870  NEXT :BP =  PEEK (AB + 1):B
    S =  PEEK (AB + 2)
1880  NEXT
1890  RETURN
1899 :
1900  REM ВЪВЕЖДАНЕ НА 16-ТИЧНО Ч
    ИСЛО
1910 RB = 0
1920  FOR I = 1 TO 2
1930  GET RA$:RA =  ASC (RA$): IF
    RA < 48 OR (RA < 65 AND RA >
    57) OR RA > 70 THEN  PRINT  CHR$
    (7);: GOTO 1930
1940  PRINT  CHR$ (RA);:RA = RA -
    48: IF RA > 9 THEN RA = RA -
    7
1950 RB = RB * 16 + RA
1960  NEXT
1970  RETURN
1999 :
2000  REM ПРОМЕНИ В СЕКТОР
2010  HOME : PRINT "П=";BP;"  С="
    ;BS;"  ПОР.С.ВЪВ ФАЙЛА=";BN:
     PRINT
```

```
2020  PRINT "МК-Я --> НАЗАД   МК-
      З --> НАПРЕД"
2030  PRINT "МК-А --> 16-ТИЧНА СТ
      ОЙНОСТ"
2040  PRINT "МК-С --> СИМВ.ОТ КЛА
      ВИАТУРАТА"
2050  PRINT "МК-Д --> ДЕСЕТИЧНА С
      ТОЙНОСТ"
2060  PRINT "МК-Р --> КРАЙ ПРОМЕН
      И И ЗАПИС"
2070  PRINT "ОСВ  --> КРАЙ ПРОМЕН
      И БЕЗ ЗАПИС"
2080  VTAB 11: POKE 34,10:RA = 0:
      BB =  - 1
2110 BB = BB + 1: IF BB > 255 THEN
      BB = 0
2120  HTAB 1: PRINT BB;: HTAB 5:R
      B =  PEEK (AB + BB): GOSUB 1
      310: HTAB 9: PRINT RB;: HTAB
      14: IF RB > 127 THEN RB = RB
      - 128
2130  IF RB > 32 THEN  PRINT  CHR$
      (RB);
2140  HTAB 17: PRINT "ИЗБОР:";
2150  GET RA$:RA =  ASC (RA$)
2160  IF RA <  > 17 GOTO 2190
2170 BB = BB - 1: IF BB < 0 THEN
      BB = 0: PRINT  CHR$ (7);: GOTO
      2150
2180  PRINT : GOTO 2120
2190  IF RA <  > 26 GOTO 2210
2200  PRINT : GOTO 2110
2210  IF RA <  > 1 GOTO 2230
2220  GOSUB 1910: PRINT : POKE AB
      + BB,RB: GOTO 2110
2230  IF RA <  > 19 GOTO 2250
2240  GET RA$: PRINT RA$: POKE AB
      + BB, ASC (RA$): GOTO 2110
2250  IF RA <  > 4 GOTO 2280
2260  INPUT "";RA: IF RA < 0 OR R
      A > 255 THEN  PRINT  CHR$ (7
      ): PRINT "ДИАПАЗОН":BB = BB -
      1: GOTO 2110
2270  POKE AB + BB,RA: GOTO 2110
2280  IF RA <  > 18 AND RA <  > 2
      7 THEN  PRINT  CHR$ (7);: GOTO
      2150
2290  IF RA = 27 THEN  PRINT : PRINT
      "КРАЯ";
2300  IF RA = 18 THEN  PRINT : PRINT
      "ЗАПИС";
2310  PRINT " СЕКТОР - ПОТВЪРДЕТЕ
      С Д:": GET RA$: PRINT : IF
      RA$ <  > "Д" THEN BB =  - 1:
      GOTO 2110
2320  IF RA = 18 THEN  GOSUB 1210

2330  POKE 34,0
2340  RETURN
2399 :
2400  REM МЕНЮ
2410  GOSUB 110: HOME : VTAB 10: HTAB
      9: PRINT "МАНИПУЛАТОР НА ФАЙ
      ЛОВЕ": VTAB 12
2420  PRINT "1 - ПЕЧАТ НА ОТДЕЛЕН
      СЕКТОР"
2425  PRINT "2 - ПОКАЗВ. НА СЕКТО
      Р - ЕКРАН"
2430  PRINT "3 - ПРОМЯНА В ОТДЕЛЕ
      Н СЕКТОР"
2440  PRINT "4 - ПЕЧАТ НА ФАЙЛ"
2445  PRINT "5 - ПОКАЗВ. НА ФАЙЛ
      - ЕКРАН"
2450  PRINT "6 - ПРОМЯНА ВЪВ ФАЙЛ
      "
2460  PRINT "7 - КРАЙ": PRINT
2470  GET RA$:RA =  ASC (RA$) - 4
      8: IF RA < 1 OR RA > 7 THEN
       PRINT  CHR$ (7);: GOTO 2470

2480  RETURN
2499 :
2500  REM ПЕЧАТ СЕКТОР
2510  PRINT D$;"PR#1":BI = 32:BJ =
      31
2515  IF RA = 2 THEN BI = 16:BJ =
      15
2520  INPUT "ПИСТА=";BP: INPUT "С
      ЕКТОР=";BS: GOSUB 1110: GOSUB
      1410
2530  PRINT D$;"PR#0"
2540  RETURN
2599 :
2600  REM ПРОМЯНА СЕКТОР
2610  INPUT "ПИСТА=";BP: INPUT "С
      ЕКТОР=";BB: GOSUB 1110: GOSUB
      2010
2620  RETURN
2699 :
2700  REM ПЕЧАТ НА ФАЙЛ
2710  PRINT D$;"PR#1":BI = 32:BJ =
      31
2715  IF RA = 5 THEN BI = 16:BJ =
      15
2720  INPUT "ИМЕ НА ФАЙЛА: ";RF$:
       GOSUB 1610: IF FI = 0 THEN
       FOR I = 1 TO 3000: NEXT : PRINT
      D$;"PR#0": RETURN
2730  GOSUB 1B10: FOR BN = 1 TO  PEEK
      (AD):BP =  PEEK (AD + (BN -
      1) * 2 + 1):BS =  PEEK (AD +
      BN * 2): GOSUB 1110: GOSUB 1
      410: NEXT
2740  PRINT D$;"PR#0"
2750  RETURN
2799 :
2800  REM ПРОМЯНА ВЪВ ФАЙЛ
2B10  INPUT "ИМЕ НА ФАЙЛА: ";RF$:
       GOSUB 1610: IF FI = 0 THEN
       FOR I = 1 TO 3000: NEXT : RETURN

2820  GOSUB 1B10: FOR BN = 1 TO  PEEK
      (AD):BP =  PEEK (AD + (BN -
      1) * 2 + 1):BS =  PEEK (AD +
      BN * 2): GOSUB 1110: GOSUB 2
      010: NEXT
2830  RETURN
```

ПАНОРАМА

УЛТРАЗВУКОВ ДИСПЛЕЙ

Учени от институтите по радиотехника и електроника и по физика при Съветската академия на науките са доказали, че някои кристали могат да светят не само под въздействието на електрически ток, рентгеново лъчение или натиск, а и от ултразвук. Едно от най-обещаващите приложения на това ново физическо явление е да се създадат плоски екраии от кристали, по които образът ще се възбужда от фокусиран ултразвуков лъч. При изпитанията се оказало, че особеио подходящи са кристалите, бракувани при производството на полупроводникови елементи: те светели хиляди пъти по-ярко от доброкачествените.

ТРИМЕРЕН ДИСПЛЕЙ

В стремежа си да увеличи обема на графичната информация върху екрана фирмата „Тектроникс" разработва тримерен дисплей, който ще използува технологията на течнокристалните индикатори. Тримерният пасивен дисплей засега няма оригинално название независимо от това, че в основата на конструкцията му лежи патентован течнокристален светлинен клапан, който представлява течнокристален панел с изменяеми размери, поставян върху екрана на традиционна електроннолъчева тръба. Очилата, които се използуват за наблюдаването на тримерното изображение, са пасивни и приличат на използуваните в стереокината.

МИНИСУПЕРКОМПЮТЪР

Досега водещата фирма в производството на суперкомпютри бе „Крей". Водеща е и цената на нейните изделия — между 10 и 15 милиона долара. В условията на остра конкуренция в играта се включи и новата фирма „Алиан" със своя минисуперкомпютър, притежаващ паралелна архитектура и бързодействие 94 милиона операции в секунда. Според мнението на специалисти това е първата ЕИМ, при която паралелната обработка се прилага автоматично при изпълнението на програма от приложен и научен характер, написани за системите Вакс на фирмата „Диджитъл икуипмънт корпорейшън". Интересът в случая е предизвикан от това, че новият минисуперкомпютър е само между 2 и 4 пъти „по-бавен" от компютрите „Крей", по затова пък се продава на цена под 1 милион долара.

АВТОМАТИЗИРАНЕ НА ИЗКУСТВЕНИЯ ИНТЕЛЕКТ

Фирмата „Айрън" е създала система за разработване на приложни програми за експертни системи з търговията, които могат да се използуват от персоналния компютър PC на фирмата ИБМ. Системата освобождава експерта по приложните задачи от процеса на непосредствено съставяне на програмите и му позволява да се съсредоточи върху създаване на правила за получаване на решения или за получаване на необходимите сведения за стоящите пред него проблеми. Програмната система се състои от две системи — за създаване и за изпълнение на приложни програми. Новият пакет притежава отворена архитектура, осигурява обединяването на вече създадените файлове и приложни програми, работи в полиекранен режим и има защита на данните чрез пароли в зависимост от желанието на потребителя.

ОТКРИТА ВЕРСИЯ НА ОПЕРАЦИОННАТА СИСТЕМА ПИК

Последната разработка на операционната система, създадена от фирмата „Пик систем", привлече вниманието на специалистите с простотата на използуването си при изпълнения на програмни средства за делови приложения. Операционната система е многообхватна, апаратно-независима и е снабдена с вградени средства за управляване на базата данни и на виртуалната памет. Откритата версия на операционната система ПИК е разработена за 16-разредни микропроцесори като персоналния компютър PC/XT на фирмата ИБМ и неговите аналози. Новата архитектура дава възможност на един компютър да се използуват едновременно няколко операционни системи, като Юникс или PC-ДОС могат да работят съвместно с ПИК.

ДВУСТРАННО ИЗПОЛЗУВАНЕ НА ДИСКЕТИТЕ

Дисковите устройства за *Правец* не са предвидени за двустранно използуване на дискетите — те имат по една записващо-четяща глава. Това обаче не пречи ка мнозина да изрязват втори опознавателен отвор на калъфката на дискетите и, като ги обръщат, да използуват и двете им страни. Двустранното използуване на този принцип е възможно, защото в повечето случаи качеството на магнитния слой от обратната страна на едностранните дискети позволява записването на информация и на двете им страни. Освен това се произвеждат и дискети, специално предназначени за двустранен запис и четене (DS/SD, DS/DD).

Все пак тази практика крие и рискове, защото при запис или четене срещуположно на главата дискетата се притиска към нея от кече, което с времето се замърсява и може да нарани непоправимо дискетата особено ако върху него са нападали твърди частици. Така че ценните си записи е най-добре да съхранявате само върху едностранно записани дискети. Добре е и да спазвате следните често пренебрегвани правила:

— дискетите да се вадят от пликовете само когато ще бъдат използувани;

— съхраняването в кутии без капаци не се препоръчва, защото прахта се насложа върху дискетите и заедно с тях попада в дисковото устройство;

— ако предназната калъфка е напрашена или зацапана, почистете дискетата, преди да я поставите в дисковото устройство.

Ето и разшифровката на крайните цифри в номерацията на дискетите според новото им означение.

EC 52 88 SS/SD
EC 52 86 DS/SD
EC 52 89 SS/DD
EC 52 87 DS/DD

SS — едностранно използуване
DS — двустранно използуване
-SD — единична плътност на записа
DD — двойна плътност на записа

ОПТИЧНО ДИСКОВО ЗАПОМНЯЩО УСТРОЙСТВО

Фирмата „Спери“ е разработила оптични дискови запомнящи устройства с повишена надеждност, които могат да се използуват в твърде неблагоприятни условия. Новата фамилия се състои от два члена — модел с капацитет 260 Мбайта и друг — със 100 Мбайта. Оптичните дискове за еднократен запис имат изключително висока надеждност в сравнение с другите носители, при които главата за запис и трене е на разстояние няколко десетки микрометри от повърхността на носителя, което създава опасност за информацията при евентуален допир на главата. Информацията от оптичните дискове се чете с лазер, който се намира на достатъчно разстояние от носителя. Новите дискове могат да се използуват съвместно и да се образува периферна памет с капацитет над 1 Гбайта, достъпът към която се осъществява от един контролер. Времето за достъп е между 100 и 200 милисекунди.

МАЛЪК ПК С ГОЛЕМИ ВЪЗМОЖНОСТИ

Фирмата AT&T е разработила персонален компютър с голяма изчислителна мощ, малки размери и възможност да работи под управлението на операционната система Юникс. Той има бързодействие 1,5 милиона операции в секунда, което е в зоната на възможности на компютрите Вакс на фирмата „Диджитъл икуипмънт корпорейшън“. Новият персонален компютър е 32-разреден, има виртуална памет, оперативна памет с капацитет 2 Мбайта, диск „Уинчестър“ с капацитет 50 Мбайта и дисково устройство — 1 Мбайт.

МУЛТИПРОЦЕСОРИ СРЕЩУ МУЛТИКОМПЮТРИ

Според дефиницията мултикомпютрите представляват мрежа от ЕИМ, обединени в общ корпус. Отделните процесори си взаимодействуват със скорост, няколко пъти по-малка, отколкото е бързодействието на паметта, и от гледна точка на програмирането тези системи създават трудности при разделяне на работата между процесорите при изпълнение на конкретна задача. Така те, въпреки че могат лесно да се адаптират към решаване на задачи с един поток инструкции и множество потоци от данни, трудно се използуват при проблеми от общ характер. Според същата дефиниция мултипроцесорите съдържат редица отделни процесори, всеки от които е включен към една и съща памет. Смята се, че такава конфигурация премахва трудностите при програмирането, които са характерни за мултикомпютрите. Фирмата „Тинкинг машинз“ вече работи по обединяването на 16 000 процесора, като крайната ѝ цел е система, съдържаща 65 000 процесора.

ХУМОР & ПРОГРАМИСТИ

Колкото и парадоксално да звучи в епохата на научно-техническата революция, най-ефективното средство за защита (и самозащита) от верижно пренапрежение и стрес е хуморът. Психолозите забелязали, че компютърните специалисти се делят на две групи. За едните всяка грешка на компютъра се превръща в лична трагедия, другите се стремят да обърнат случилото се в шега. Затова психолозите най-сериозно и най-настойчиво съветват хората без чувство за хумор да не стават програмисти.

4 МБИТОВА ПАМЕТ ВЪРХУ ЕДИН ЧИП

Динамичните запомнящи устройства с произволен достъп и капацитет 1 Мбита вече са реалност. Специалистите по полупроводникова техника вече говорят за памети с капацитет 4 Мбита върху един чип. Песимистите обаче са на мнение, че тези памети изискват по-ниски проектни норми и по-сложни архитектурни решения, които са над възможностите на съвременната технология. И въпреки всичко компанията „Мицибиши“ съобщава за инсталация за обработка на материалите, с която може да се получи необходимата плътност на елементите. Инсталацията работи по така наречения метод на плазмено гравиране с електронно-циклотронен резонанс. Досегашната технология използуваше реактивното йонно гравиране, при което йоните непосредствено удрят върху повърхността на кристала през прозорците на маската, рисувайки необходимата схема. Така се формират линии с дебелина до 1 микрометър. Новото производство обаче изисква субмикронни топологични структури. За споменатата система само се казва, че пропуска свръхвисокочестотен лъч през електромагнитно поле, получено от бобини, обхващащи отвън камерата за гравиране. По този начин могат да се получат линии с ширина до 0,2 микрометра. Мегабитовите памети се очакват на пазара след две-три години.

СТРАННАТА БЪТЕРФЛАЙ

Разработеният от фирмата BBN компютър, наречен Бътерфлай, обединява 128 процесора и изпълнява над 60 милиона инструкции в секунда. Той представлява паралелен процесор, мултипроцесор, мултикомпютър и на практика нито едно от горепосочените устройства. Специалистите от фирмата първоначално го характеризирали като микропроцесорна система с тясно взаимодействие. За паралелна работа на процесорите се използува превключваща система, чиито възможности нарастват с увеличаване на броя на процесорите. Всеки процесор увеличава производителността на компютъра с около половин милион инструкции в секунда. Обемът на общата за системата памет също нараства пропорционално. В случая е използуван мултипроцесорният подход (много процесори с обща памет), а не мултикомпютърният, при който за всеки процесор има отделна памет. Така се избягват проблемите при програмирането, което на практика се доближава до това при обикновените компютри.

СЪДЪРЖАНИЕ

СУПЕРМИНИ-КОНКУРС „10—20“

Програмата на Владимир Крумов, ученик от образцовия техникум "С. М. Киров" – София върви следната полезна работа. След стартирането програмата "обръща" клавиатурата на Правец 82 – сменя местата на кирилицата и латиницата. След всяка команда &ё те отново си сменят местата. Видът на основната клавиатура се индикира с L или K в маркера. След многократни промени първоначалната клавиатура се възстановява чрез RESET.

```
0 REM ВЛАДИМИР КРУМОВ
10 FOR A = 768 TO 830: READ B: POKE
   A,B: NEXT : TEXT : HOME : VTAB
   12: PRINT "В МОМЕНТА ОСНОВНИ
   ЯТ РЕГИСТЪР Е ЛАТИНИЦА": POKE
   1013,76: POKE 1014,0: POKE 1
   015,3: CALL 768
20 DATA 169,11,133,56,169,3,133
   57,76,234,3,32,27,253,201,21
   9,208,3,169,133,96,201,221,2
   08,3,169,144,96,201,128,208,
   15,164,36,173,61,3,73,7,141,
   62,3,145,40,76,11,3,201,192,
   144,10,72,173,62,3,73,104,17
   6,2,73,32,96,76
```


КОМПЮТЪР ЗА ВАС
Издание на ЦК на ДКМС

СПИСВА РЕДАКЦИЯТА НА ВЕСТНИК „ОРБИТА“

Главен редактор
д-р ДИМИТЪР ПЕЕВ _88-51-68_

1000 София СОФИЯ
БУЛ. „ТОЛБУХИН“ № 51 А
ТЕЛ. 87-78-04

Приемни часове от 14 до 16 ч.

ЗАМ. ГЛАВЕН РЕДАКТОР
И ЗАВ. СПИСАНИЕТО
инж. Георги Балански 87-09-14

ОТГОВОРЕН СЕКРЕТАР
инж. Борис Аиков 80-23-18

ДЕЖУРЕН РЕДАКТОР
Кънчо Кожухаров

ДИЗАЙНЕР
Васил Пендев

КОРЕКТОР
Бистра Ботева

ТЕХНИЧЕСКИ РЕДАКТОР
Люба Калайчиева

Предадено за печат
20 февруари 1986 г.

Подписано за печат
26 март 1986 г.

Печатни коли 6

Формат 60/90/8

Тираж 20 000

Цена 1,20 лв.
Годишен абонамент 7,20 лв.

ДП "Д. Благоев"
София, ул. "Ракитин" 2
Телефон 46-31

Индекс 20 593

НЕПУБЛИКУВАНИ РЪКОПИСИ И ПРОГРАМИ НЕ СЕ ВРЪЩАТ.

РЕДАКЦИОНЕН СЪВЕТ: чл.-кор. Ангел Ангелов, проф. Ангел Писарев, ст.н.с. к.т.н. инж. Александър Александров, академик Благовест Сендов, Веселин Спиридонов, доц. Димитър Шишков, инж. Иван Маранеозов, инж. Пенчо Сираков, чл. кор. Петър Кендеров, н.с. к.т.н. инж. Пламен Вачков, Рашко Ангелинов, инж. Иван Михайлов

(2A-2B)	(42-43)	BAS2L BAS2H	Указател на адреса на следващия ред при смяна на редовете в текстов режим (BASE POINTER; L - младши байт, H - старши байт).
2C	44	H2	Крайна дясна точка на хоризонталната линия, изчертавана с командата HLINE (HORIZONTAL LINE).
2D	45	V2	Крайна долна точка на вертикалната линия, изчертавана с командата VLINE (VERTICAL LINE).
(2C-2D)	(44-45)	LMNEM RMNEM	Указател на адреса на мнемоничния код при реасемблиране (MNEMONIC ADDRESS POINTER).
2E	46	CHKSUM	Натрупва контролната сума при четене на данни от касета (CHECK-SUM).
2E	46	FORMAT	Специфицира формата на инструкцията за извеждане върху екрана при реасемблиране.
2E	46	MASK	Маска за цвета в графичен режим с ниска разделителна способност.
2F	47	LASTIN	Следи промяната на входа при четене на данни от касета.
2F	47	LENGHT	Дължина на инструкцията при реасемблиране.
*2F	47	SIGN	Знак на резултата при умножение и деление на двубайтови числа (подпрограми MULPM и DIVPM) в бит 1.
30	48	COLOR	Код на цвета за графичен режим с ниска разделителна способност.
31	49	MODE	Показва разположението на шестнайсетичната информация в извеждания ред при изпълнение на командите на Монитора.
32	50	INVFLG	Съдържанието на тази клетка определя вида на знаците в текстов режим - 255($FF) = NORMAL; 127($7F) = FLASH; 63($3F) = INVERSE (INVERSE FLAG).
33	51	PROMPT	Съдържа кода на знака на курсора.
34	52	YSAV	Съхранява съдържанието на регистър Y при изпълнение на командите на Монитора (Y SAVE).
35	53	YSAV1	Съхранява съдържанието на регистър Y при извикване на подпрограмите за извеждане на екрана.
(36-37)	(54-55)	CSWL CSWH	Формират изходният регистър на куплунгите (SLOTS) на компютъра. RST , 0 MK-P (CTRL-P) и PR #0 установяват в тези клетки адрес $FDF0 (подпрограма за извеждане върху екрана - COUT1). S MK-P и PR #S установяват в тези клетки адрес $CS00, където S е номерът на куплунга (CHARACTER OUTPUT SWITCH; L - младши байт, H - старши байт).

СЛЕДВА

младия човек, най-вече на неговото творческо мислене. Той сам създава системата, написва програмите за нейното управление, прави пробн, изнася корекции и така проследява целия процес, използувайки нагледни, прости, безопасни и сравнително евтини учебни средства.

Минироботите и учебните ГАПС се разпространяват от ЦК на ДКМС — ИВСД „Авангард“ и МНП.

◀ Представители на фамилията Робко

▼ Робко-111, управляван от Правец-8М

ФАМИЛИЯТА „РОБКО“

Фамилията РОБКО от учебни роботи и средства за изграждане на учебни ГАПС, разработвана в ИТКР—БАН със съдействието на ИВСД „Авангард“, цели заедно с втората компютърна грамотност нашата младеж да получи и грамотност по роботика и ГАПС. Персоналните компютри и учебната роботика се допълват взаимно много удачно. Учебните роботи и средствата на изграждане на учебни ГАПС се управляват и програмират с осемразредните персонални компютри от фамилията Правец. Те дават възможност постепенно с нарастваща сложност и при активно участие на обучаемия да се развиват способностите на