МИКРОКОМПЮТЪРНА ТЕХНИКА ЗА ВСИЧКИ

КОМПЮТЪРЪТ ИГРАЕ, РИСУВА И СВИРИ

9. КОМПЮТЪРЪТ ИГРАЕ, РИСУВА И СВИРИ

ПОРЕДИЦАТА Е СЪЗДАДЕНА
СЪС СЪДЕЙСТВИЕТО
НА ЦЕНТРАЛНИЯ КОМИТЕТ НА ДКМС
И МИНИСТЕРСТВОТО НА НАРОДНАТА ПРОСВЕТА

К.т.н. инж. ВЕЛЧО С. ВЕЛЕВ
К.т.н. инж. НАДЕЖДА К. ПЕТРОВА

КОМПЮТЪРЪТ ИГРАЕ, РИСУВА И СВИРИ

ДЪРЖАВНО ИЗДАТЕЛСТВО „ТЕХНИКА“
СОФИЯ, 1987

УДК 681.3.06

В книгата са описани техническите и програмните възможности на микрокомпютъра Правец-82 за осъществяване на графични изображения, музика и игри. Разгледани са основните средства и похвати за решаването на тези задачи. Изложението е подкрепено с много примерни програми, които са записани на приложената към книгата дискета. В дискетата са включени и няколко програми-редактори.
Книгата е предназначена за всички читатели, които са запознати с програмирането на БЕЙСИК за Правец-82 и желаят да доразвият и задълбочат знанията си по изложените въпроси.

681

ПРЕДГОВОР

Някога Аристотел е чертаел фигурите си на пясъка. Дедите ни изпълняваха музикални мелодии на обикновена дървена свирка. Децата от нашето поколение се радваха на забавните игри „Домино" или „Не се сърди човече".

Днес в ерата на научно-техническата революция за никого не е чудно, че електронноизчислителните машини конструират сложни съоръжения, композират и изпълняват музика, моделират многовариантни игрови ситуации.

Въпреки това появата на микрокомпютрите предизвика изненада дори у специалистите. Те се събират върху част от бюрото ни, а предоставят изчислителна мощност и възможности, които до преди няколко години бяха привилегия само на малък брой програмисти, работещи в модерни изчислителни центрове.

Нашата страна реагира бързо на това постижение на техническия прогрес. Талантливи конструктори създадоха български микрокомпютри. Започна производството на ИМКО—2 и Правец-82, които се разпространиха бързо. Те предизвикаха неподозирано широк интерес сред подрастващите, младежта и възрастните.

Ако се опитаме да потърсим причините за този интерес, трябва да отбележим следните фактори:

широкото разпространение на микрокомпютрите;

опростената работа с тях;

възможността им да обменят информация с потребителя чрез текст, образи и звуци;

наличието на множество компютърни игри за тях.

Използуването на микрокомпютрите за развлекателни игри предизвика незабавно критични мнения, но то има и свои защитници. Вероятно спорът ще продължи дълго. Засега всички са съгласни най-малко с две неща. Първо, че компютърните игри доставят удоволствие, и второ, че те са подходяща форма за преодоляване на бариерата между неспециалиста по изчислителна техника и съвременното чудо на техниката — микрокомпютъра.

Безспорно някои „потребители" на микрокомпютрите, които са привлечени от възможностите им за игри, ги използуват само за развлечение. Друга част обаче, особено подрастващите, се интересува силно какво се крие зад тези игри, как са осъществени и как могат да създадат сами програми за игри. Това вече

не е малко. У тази група потребители е предизвикан интерес към тайните на електрониката, изчислителната техника и програмирането. Тъй като да се състави програма за хубава игра, не е проста задача; книгата има за цел да помогне именно на тази категория читатели.

Да се програмират игри за компютри не е нова дейност. Тя започна с разпространяването на големите изчислителни машини, разшири се с появата на миникомпютрите и доби масови мащаби с усвояването на микрокомпютърната техника. Сега в световен мащаб има множество микрокомпютри, които разполагат със специални възможности за игри. За да се обяснят ефективно идеите и програмните похвати за програмиране на игри, необходимо е да се съсредоточи вниманието върху един компютър. Тук е избран микрокомпютърът Правец-82.

В книгата на достъпно ниво са описани графичните и звуковите възможности на Правец-82, както и програмните похвати и идеи, които са използувани при осъществяване на най-добрите игри, създадени за този микрокомпютър. Засегнати са въпросите за осъществяване на графични и текстови изображения в графичен режим с ниска и висока разделителна способност, управление на цветовете, осъществяване на анимация, откриване на сблъсквания и произвеждане на звуци. Разглежданията са направени на основата на БЕЙСИК, но за някои от тях има разширения за АСЕМБЛЕР и машинен език.

Книгата не е предназначена за начинаещи програмисти. Тя изисква предварителни знания по БЕЙСИК. Ако не знаете версията на БЕЙСИК за Правец-82, необходимо е преди да започнете четенето на тази книга, да се насочите към другите книги от поредицата. Освен това за успешното усвояване на голяма част от изложените въпроси е необходимо да познавате и да умеете да работите с управляващата програма МОНИТОР. За изучаването ѝ може да ползувате първата книга от поредицата.

ГЛАВА 1. ОСОБЕНОСТИ НА ПРАВЕЦ-82 ПРИ ОСЪЩЕСТВЯВАНЕ НА ГРАФИЧНИ ИЗОБРАЖЕНИЯ И ИГРИ

Версията на БЕЙСИК за микрокомпютъра Правец-82 съдържа разширения на стандартните средства на езика, ориентирани към създаването на графични изображения и игри.Такива са например командите GR, HGR, HLIN и др., с които се чертаят графични фигури. Те са полезни, прости и удобни за работа, но в някои случаи се оказват недостатъчно бързи или с ограничени възможности за достъп до паметта. Тогава се използуват други средства на езика, например PEEK, POKE, CALL или USR, които преодоляват тези недостатъци. С възможностите си за непосредствен достъп до паметта и извикване на машинни програми те могат да заместят практически всяка команда на БЕЙСИК или да разширят значително функциите му. Затова ще разгледаме особеностите на тези средства на езика, свързани с тематиката на книгата.

Микрокомпютърът Правец-82 съдържа електронни схеми, които извеждат информацията за изображението от определени области на паметта върху екрана на видеомонитора. В зависимост от режима, в който се намират, те извеждат данни от една или друга област и ги интерпретират по различен начин. Има *три основни режима за извеждане на данни върху екрана:* текстов режим, графичен режим с ниска разделителна способност и графичен режим с висока разделителна способност. В тази глава се разглеждат особеностите на различните режими и начините, по които те се избират.

1.1. ФУНКЦИЯ PEEK

Чрез функцията PEEK можем да „надникнем" в която и да е клетка от паметта и да проверим нейното съдържание. При това проверката се извършва от програма на БЕЙСИК, без да е необходимо да се вика машинна програма или да се влиза в управляващата програма МОНИТОР. Например с A = PEEK (222) се извлича съдържанието на клетката с адрес 222 и се присвоява на променливата A. Функцията PEEK е ориентирана десетично, т.е. както зададеният адрес, така и извлечената стойност, са десетични числа.

Тя има разнообразно приложение. Освен за проверка на съдържанието на клетки от паметта с нея може да се установи дали даден клавиш е бил натиснат. При много игри, докато се осъществява движение на фигури по екрана, трябва да се проверява дали не е натиснат клавиш от клавиатурата или не се е изменило състоянието на игровия контролер. С функцията PEEK може да се следи съдържанието на определени адреси от паметта, в които се съхранява служебна информация на компютъра, важна за правилното изпълнение на потребителските програми.

Използуването на функцията PEEK се илюстрира добре в програма 1.1. След като въведете и стартирате тази програма, ще видите, че буквата M се изписва непрекъснато върху екрана. Ако натиснете някакъв клавиш, съответният знак ще започне да се изписва, докато не натиснете друг клавиш и т.н. Изпълнението на програмата ще спре, ако натиснете клавиша RST.

```
 1  REM **ПРОГРАМА1.1**
 2  REM **ФУНКЦИЯ PEEK**
10  REM *НАЧАЛНА СТОЙНОСТ ЗА F$*
20  F$ = "M"
30  REM *ИЗВЕЖДАНЕ НА F$*
40  PRINT F$;
50  L = PEEK (49152)
60  IF L > 127 THEN  CALL  - 1059:
    F$ =  CHR$ (L - 128): POKE - 16368,0
70  GOTO 30
```

С ред 20 в програмата се присвоява начална стойност M на символната променлива F$, а с ред 40 съдържанието ѝ се извежда върху екрана. Символът ; потиска автоматичното връщане на печата на нов ред. След това се извлича съдържанието на клетка с адрес 49152 и се присвоява на променливата L (ред 50). Тази клетка съдържа информация за две неща. Ако най-левият ѝ бит е 1, т.е. съдържанието ѝ е по-голямо или равно на 128, това означава, че е натиснат някой от клавишите. Тогава останалите битове съдържат кода на символа, който се въвежда с този клавиш. Затова с ред 60 от програмата се проверява дали променливата L е по-голяма от 127. Ако това е вярно, с командата CALL се извършва обръщение към адрес −1059, с което се включва високоговорителят и се чува кратък звуков сигнал. След това на променливата F$ се присвоява новата стойност и в клетка −16368 (49168) се зарежда нула. Така клавиатурата се подготвя за натискане на следващ клавиш. В ред 70 се затваря цикъл, който продължава извеждането на съдържанието на променливата F$.

При програмирането на игри, които се управляват с игрови контролери, функцията PEEK се използува за приемане на сигна-

лите от натискането на бутоните на контролерите. Ако например след изпълнение на X = PEEK (−16287) X е по-голямо от 127, това означава, че е бил натиснат бутонът на игровия контролер с номер 0. Ако X = PEEK(−16286) и X е по-голямо от 127, бил е натиснат бутонът на игровия контролер с номер 1.

1.2.КОМАНДА POKE

Командата POKE позволява да се записват стойности в отделни клетки на паметта от програма на БЕЙСИК. Например POKE 36,5 записва числото 5 в клетка с адрес 36. Запомнете, че подобно на функцията PEEK, тази команда е ориентирана десетично , т.е. адресът и стойността, която искаме да запишем, трябва да са представени в десетична бройна система. Съдържанието на клетка 36 определя хоризонталната позиция на маркера. Поради това POKE 36,5 има същото действие, както НТАВ 5.

```
  1  REM **ПРОГРАМА 1.2**
  2  REM **КОМАНДА POKE**
 10  REM *РЕЖИМ ВРС*
 20  HOME : HGR
 30  FOR I = 1 TO 200
 40  A% = ( RND (1) * 8192) + 8192
 60  POKE A%,1: REM 1 В ИЗБРАНИЯ АДРЕС
 70  NEXT
 80  FOR I = 1 TO 10000: NEXT
 90  TEXT : REM  ТЕКСТОВ РЕЖИМ
100  END
```

Програма 1.2 демонстрира приложение на командата POKE, имащо пряко отношение към графичните изображения. В ред 40 произволно се избират адреси от областта на паметта (8192 — 16384), която съдържа данни за изображенията от първа графична страница в режим на висока разделителна способност. Чрез командата POKE в тези адреси се записва 1. Това води до появата на светла точка върху екрана, тъй като в ред 20 програмата е установила графичен режим с висока разделителна способност. Цикълът се повтаря 200 пъти и на екрана се изчертават 200 точки, наподобяващи звезди върху ясно небе.

1.3. КОМАНДА CALL

Командата CALL действува подобно на командата GOSUB. Разликата е, че вместо подпрограма на БЕЙСИК се стартира под-

програма на машинен език. След нейното изпълнение управлението се връща в главната програма непосредствено след командата CALL. Извикването на програми на машинен език дава допълнителни възможности за работа с компютъра и повишава скоростта на изпълнение на критични участъци от програмите на БЕЙСИК. При програмирането на игри често е невъзможно да се постигне плавно движение на фигурите върху екрана и е трудно да се възпроизведе подходящ звук или мелодия, ако се разчита само на програма на БЕЙСИК.

Най-простият начин да използуваме командата CALL е да извикваме машинни програми, записани в паметта на компютъра при производството му. Например изпълнението на командата CALL -936 изчиства текстовия екран и премества маркера в горния ляв ъгъл. Същият резултат се получава, ако зададем командата HOME, защото при изпълнението на HOME се извиква същата машинна програма. Аналогично командата CALL -384 включва инверсен режим, а CALL -380 възстановява нормален режим на изображението върху екрана. Съществуват много други програми на машинен език, записани в паметта ROM, които могат да се извикат чрез командата CALL.

Другите начини за използуване на командата CALL изискват програмата на машинен език да се зареди предварително в паметта на компютъра. Тя се прочита или от външен носител (с команда BLOAD), или от програма на БЕЙСИК (с команда READ). В програмата на БЕЙСИК машинната програма се записва като последователност от константи с една или повече команди DATA. Този начин е показан в програма 1.3. Машинната програма е записана в ред 30. Числовите константи се четат последователно с командата READ и се записват в паметта на компютъра от адрес 768. Когато се прочетат всички данни, чрез командата CALL управлението се предава към адрес 768, където вече е записана машинната програма. Тя задейства с подходяща честота високоговорителя на микрокомпютъра, в резултат на което се възпроизвежда звук, подобен на сирена. Продължителността на звука се определя от съдържанието на клетката с адрес $7 (7), която в ред 20 зареждаме със 100.

```
 1  REM **ПРОГРАМА 1.3**
 2  REM **КОМАНДА CALL**
 5  REM *ЗАРЕЖДАНЕ НА МАШИННАТА ПРОГРАМА*
10  FOR I = 1 TO 19: READ W: POKE  I + 767,W: NEXT
15  REM *ИЗПЪЛНЕНИЕ*
20  POKE 7,100: CALL 768
25  REM *МАШИННА ПРОГРАМА*
30  DATA 173, 48, 192, 136, 208, 4, 198, 7, 240, 8, 202, 208, 246,
    166, 7, 76, 0, 3, 96
40  GOTO 20
```

По описания начин в паметта на микрокомпютъра може да се зарежда всяка машинна програма и след това да се извиква за изпълнение от програма на БЕЙСИК чрез командата CALL.

1.4.ФУНКЦИЯ USR

Тази функция също се използува за стартиране на машинна програма.Разликата е, че когато използуваме USR, можем да прехвърлим до два байта информация от главната към извиканата програма. След изпълнението на машинната програма тя също може да върне стойност към главната програма.

Функцията USR се използува само като операнд на друга команда от БЕЙСИК, например X = USR(2) или PRINT USR(4). Стойността, зададена в скоби, е десетично число, което се предава на машинната програма. Числото, което машинната програма връща след изпълнение, се поставя на мястото на USR в програмата на БЕЙСИК.

```
 1  REM **ПРОГРАМА 1.4**
 2  REM **ФУНКЦИЯ USR **
10  FOR I = 1 TO 26: READ W: POKE
    I + 767,W: NEXT : REM  ЗАРЕЖ-
    ДАНЕ НА МАШИННАТА ПРОГРАМА
14  PRINT "ВЪВЕДЕТЕ СТОЙНОСТ ОТ 1 ДО 255"
16  INPUT C$: C  = VAL (C$)
18  POKE 10,76: POKE 11,0: POKE 12,3
20  A = USR (C): REM ИЗПЪЛНЕНИЕ
24  DATA 32,12,225
26  DATA 165,161
28  DATA 133,7
29  REM *МАШИННА ПРОГРАМА*
30  DATA 173, 48, 192, 136, 208, 4, 198, 7, 240, 8,
    202, 208, 246, 166, 7, 76,  7, 3, 96
40  GOTO 14
```

Действието на функцията USR е показано в програма 1.4. Тя е изменен вариант на програмата 1.3. Вместо CALL в ред 20 е използувана функцията USR. Чрез нея основната програма предава информация на машинната програма за височината на произвеждания звук. Тази информация се задава от потребителя чрез редове 14 и 16.

Изпълнението на функцията USR изисква допълнителна подготовка. Редове 24, 26 и 28 съдържат машинни инструкции, които се добавят в началото на стартираната машинна програма. Ред 24 включва машинна инструкция JSR $E10C, която предава

управлението към стандартна машинна програма. Тя прехвърля в адреси $A0 (160) и $A1 (161) стойността, която се подава към машинната програма чрез функцията USR. В редове 26 и 28 има две машинни инструкции, които преместват съдържанието на адрес $A1 (161) в адрес $7 (7). Това е мястото, където за разглежданата машинна програма се съдържа стойността, която определя височината на произвеждания звук. Тъй като тази стойност е в границите от 0 до 255 (вж.ред 14), старшата част на стойността, която се предава чрез USR, записана на адрес $A0, не се обработва.

С командите POKE от ред 18 се зарежда инструкция JMP $300 в адреси от $0A до $0C.Тя представлява безусловен преход към адреса, съдържащ началото на машинната програма, която искаме да стартираме чрез USR. За разглеждания пример този адрес е $300(768). За други случаи той може да е друг.Съществено е да се знае, че в клетки с адреси от $0A до $0C предварително трябва да е записана инструкцията JMP заедно с адреса, от който започва машинната програма.

1.5. ФУНКЦИЯ PDL

Към микрокомпютъра Правец-82 могат да се свържат до четири контролера за игри. Те съдържат резистори, чиито съпротивления могат да се променят в границите от 0 до 150 Ω чрез завъртане или преместване на ръчка. Стойността на съпротивлението се проверява от електронна схема, която чрез функцията PDL подава информация за положението на ръчката към програма на БЕЙСИК.

Функцията PDL има формат PDL (X). Променливата X съдържа номера на контролера, от който се приема информация и трябва да е между 0 и 3 включително. Стойността, която функцията подава към програмата на БЕЙСИК, е в границите от 0 до 255 и е пропорционална на текущата стойност на съпротивлението на контролера.

1.6.ТЕКСТОВ РЕЖИМ

В текстов режим микрокомпютърът може да извежда върху екрана до 24 реда текст. Всеки ред съдържа до 40 знака.

Съществуват две области от паметта, информацията от които може да се изведе като текст върху екрана. Те се наричат първа и втора текстова страница. Разположението им се вижда от картата за разпределение на паметта, съдържаща се в табл.1.1.

Таблица 1.1

Област от паметта и предназначение	Начален адрес		Краен адрес	
	Шестн.	Десетичен	Шестн.	Десетичен
МОНИТОР	$F800	63488	$FFFF	65535
БЕЙСИК	$E000	57344	$F7FF	63487
Запазена	$D000	53248	$DFFF	57343
Вход/Изход	$C000	49152	$CFFF	53247
ДОС	$9600	38400	$BFFF	49151
Свободна	$6000	24576	$95FF	38399
ВРС, втора страница	$4000	16384	$5FFF	24575
ВРС, първа страница	$2000	8192	$3FFF	16383
Свободна	$C00	3072	$1FFF	8191
ТЕКСТ/НРС, втора страница	$800	2048	$BFF	3071
ТЕКСТ/НРС, първа страница	$400	1024	$7FF	2047
Вектори на ДОС	$3CO	960	$3FF	1023
Свободна	$300	768	$3BF	959
Входен текстов буфер	$200	512	$2FF	767
Стек на микропроцесора	$100	256	$1FF	511
Нулева страница	$0	0	$FF	255

Командата TEXT извежда автоматично върху екрана първа текстова страница. Няма команда на БЕЙСИК, която извежда втората текстова страница. Една от причините за това е, че областта от паметта, която се използува за втората текстова страница, се използува и от програмите на БЕЙСИК. Поради това с втората текстова страница можем да работим само когато програмираме на асемблерен или машинен език.

В текстовия режим всеки знак, който се извежда върху екрана, се представя в паметта на микрокомпютъра с определен числов код. За други нужди символите се кодират с кодовете, представени в приложение 1. Имайте предвид, че вътрешното кодиране на знаците за нуждите на текстовото изображение се различава от кодирането им, показано в приложение 1.

1.7. ГРАФИЧЕН РЕЖИМ С НИСКА РАЗДЕЛИТЕЛНА СПОСОБНОСТ

Режимът с ниска разделителна способност (НРС) също използува две области от паметта, наречени съответно първа и втора графична страница за режим НРС. Информацията от всяка страница може да се извежда върху екрана в пълен графичен режим или в смесен режим (текст и графика). Графичният режим НРС използува същите физични области от паметта, които използува и текстовият режим (табл.1.1). Втората страница на режим НРС съвпада с областта, която се използува от програмите на БЕЙСИК, и има ограничено приложение.

Командата GR установява автоматично смесен режим НРС и първа графична страница. Няма команда от БЕЙСИК, която установява пряко пълен графичен режим НРС и втора графична страница.

1.8. ГРАФИЧЕН РЕЖИМ С ВИСОКА РАЗДЕЛИТЕЛНА СПОСОБНОСТ

Режимът с висока разделителна способност (ВРС) също има първа и втора графична страница. Съдържанието на всяка от тях може да се изведе върху екрана в пълен графичен или смесен (с четири реда текст) режим. Командата HGR установява смесен режим ВРС с изображение от първата графична страница. Командата HGR2 установява пълен графичен режим ВРС с изображение от втората графична страница. Няма команди от БЕЙСИК, които установяват пряко останалите разновидности на режим ВРС.

1.9. ПРОГРАМНИ КЛЮЧОВЕ

Версията на БЕЙСИК за микрокомпютъра Правец-82 разполага с полезни команди за работа в графичен режим, но те имат известни недостатъци. С тях не може да се установи пряко пълен графичен режим ВРС с изображение от първа графична страница, смесен режим ВРС с изображение от втора графична страница или превключване между двете графични страници, без да се разруши съдържащата се в тях информация. Всяка команда от БЕЙСИК за работа в графичен режим предизвиква няколко действия. Например командата HGR.

установява графичен режим ВРС с изображение от първа графична страница;

изчиства първа графична страница;

определя първа графична страница като област от паметта,

в която ще се записва информацията при следващо изпълнение на команди за чертане.

Има случаи, в които искаме да изведем съдържанието на графичните страници, без да ги изтриваме. Специални ефекти при програмиране на движещи се фигури можем да постигнем, ако извеждаме съдържанието на едната графична страница, докато чертаем в другата, и след това извеждаме съдържанието на втората страница, докато чертаем в първата. Ясно е, че тези изисквания не можем да удовлетворим само чрез командите HGR и HGR2. Затова често желаният графичен режим се установява, като се използуват т.нар. *програмни ключове*.

Четирите ключа, които определят режимите за извеждане на изображения върху екрана, се наричат програмни по две причини. Първо, за да се отбележи, че не са механични, и второ, защото те могат да се превключват от системните програми на микрокомпютъра или от програми, които ние сме написали.

Удобно е за по-нататъшните обяснения да си представяме тези ключове като двупозиционни, т.е., че всеки ключ има две положения, съответствуващи на двете състояния на режима, който управлява. Първият ключ избира графичен или текстов режим; вторият – пълен графичен или смесен режим; третият – първа или втора страница; четвъртият – графичен режим HPC или BPC.

Следователно всеки ключ управлява само една характеристика от процеса на извеждане на изображението. Командите на БЕЙСИК превключват няколко ключа едновременно. Например HGR установява първия ключ в графичен режим, втория – в режим на смесен екран, третия – на първа графична страница и четвъртия – в режим BPC. Командата GR се различава по това, че установява четвъртия ключ в режим HPC.

Всеки ключ може да се установи в една от двете му позиции чрез обръщение към определен адрес. За да установим първия ключ в графичен режим, можем да се обърнем с функцията PEEK към адрес 49232 (–16304) или да запишем нова стойност в същия адрес чрез командата POKE. Величината, която ще прочетем или запишем, в случая няма никакво значение. Важно е обръщението към съответния адрес. Например всяка от следните команди: POKE 49232,175; X = PEEK (49232) и PRINT PEEK (49232) ще установи първия ключ в графичен режим. Ако сме в режим на програмата МОНИТОР, можем да установим графичен режим с командата: * C050. В случая ще се изведе съдържанието на клетка с адрес $C050 (49232), т.е. ще се извърши действително обръщение към този адрес. Същия ефект ще постигнем, ако запишем 0 в клетка с адрес $C050, като използуваме командата * C050: 0. И тук съществено е да имаме обръщение към съответния адрес, а не е важно дали ще четем или ще записваме и какво ще запишем.

Адресите, към които трябва да се обръщаме, за да установим всяко от двете положения на четирите ключа, са дадени в табл.1.2.

Таблица 1.2

Ключ	Позиция	Команда
1	графичен режим	POKE −16304,0
	текстов режим	POKE −16303,0
2	пълен екран	POKE −16302,0
	смесен екран	POKE −16301,0
3	първа страница	POKE −16300,0
	втора страница	POKE −16299,0
4	НРС	POKE −16298,0
	ВРС	POKE −16297,0

Всеки ключ остава непроменен, докато не бъде превключен (чрез обръщение към съответния адрес) в другото положение.

Важно е, че четирите ключа се управляват независимо един от друг. На тази основа можем да стигнем до извода, че има 16 различни режима за извеждане на изображението. Това обаче не е вярно. Когато ключ 1 е в положение на текст (POKE −16303,0), промяната на ключове 2 и 4 остава без последствие.

Възможни са общо 10 различни режима за извеждане на изображението, дадени в табл. 1.3 заедно с комбинацията от адреси, към които трябва да извършим обръщение, за да установим всеки от тях.

Таблица 1.3

Режим	Команди
НРС, първа страница пълен екран	POKE −16298,0; POKE −16304,0 POKE −16300,0; POKE −16302,0
НРС, първа страница смесен екран	POKE −16298,0; POKE −16304,0 POKE −16300,0; POKE −16301,0
НРС, втора страница пълен екран	POKE −16298,0; POKE −16304,0 POKE −16299,0; POKE −16302,0
НРС, втора страница смесен екран	POKE −16298,0; POKE −16304,0 POKE −16299,0; POKE −16301,0
ВРС, първа страница пълен екран	POKE −16297,0; POKE −16304,0 POKE −16300,0; POKE −16302,0

Таблица 1.3 продължение

Режим	Команди
ВРС, първа страница смесен екран	POKE −16297,0; POKE −16304,0 POKE −16300,0; POKE −16301,0
ВРС, втора страница пълен екран	POKE −16297,0; POKE −16304,0 POKE −16299,0; POKE −16302,0
ВРС, втора страница смесен екран	POKE −16297,0; POKE −16304,0 POKE −16299,0; POKE −16301,0
ТЕКСТ, първа страница	POKE −16303,0; POKE −16300,0
ТЕКСТ, втора страница	POKE −16303,0; POKE −16299,0

От един режим в друг се преминава, като се променят само ключовете, които трябва да заемат ново положение. Например, за да преминем от режим на пълен екран, ВРС, първа графична страница в режим на смесен екран, ВРС, първа графична страница е достатъчно да превключим само ключ 2 чрез обръщението POKE −16301,0. За да променим режима смесен екран, ВРС, първа страница в режим смесен екран, ВРС, втора графична страница трябва да превключим само ключ 3 чрез POKE −16299,0.

Използуването на програмните ключове е показано в програмите 1.5 и 1.6.

```
 1  REM **ПРОГРАМА 1.5**
 2  REM **ПРИМЕР 1 – ПРОГРАМНИ КЛЮЧОВЕ**
 5  REM *ВРС ПЪРВА СТРАНИЦА*
10  HGR : HCOLOR= 3
20  HPLOT 80,100 TO 200,100
30  HGR2 : REM *ВРС ВТОРА СТРАНИЦА*
40  HPLOT 90,40 TO 190,160:
    HPLOT190,40 TO 90,160
50  FOR I = 1 TO 100
60  FOR J = 1 TO 100: NEXT J
70  A = 1 - A
80  POKE - 16299 - A,0
90  NEXT I
```

В програма 1.5 отначало се начертават две различни фигури в първа и втора графична страница на режим ВРС. Двете страници на режим ВРС се избират в редове 10 и 30 чрез командите HGR и HGR2. Съответните фигури се изчертават в редове 20 и 40 чрез HPLOT. След това започва цикъл (редове 50 – 90), с който се извежда последователно върху екрана съдържанието на първа и втора графична страница. Двете страници се превк-

лючват чрез редуващи се обръщения към адреси −16300 и −16299, които променят положението на ключ 3. За да се задържат изображенията достатъчно дълго време върху екрана, на ред 60 е програмиран вътрешен цикъл, с който се увеличава времето между две превключвания на първа и втора страница.

Програма 1.6 извежда върху екрана движещи се квадрати. Отначало те нарастват, а след това намаляват.

```
  1  REM **ПРОГРАМА 1.6**
  2  REM **ПРИМЕР 2 – ПРОГРАМНИ КЛЮЧОВЕ**
 10  HGR : HGR2
 20  A = 1:B =  - 1:C = 2:D = 1
 30  FOR I = 1 TO 30
 40  FOR J = 1 TO 2
 45  REM *ИЗБОР НА СТРАНИЦА ЗА ЧЕРТАНЕ*
 50  POKE 230,32 * J
 60  A = A + B
 70  HCOLOR= 0
 80  HPLOT 140 - A,96 - A TO 140 + A,
     96 - A TO 140 + A,96 + A TO
     140 - A,96 + A TO 140 - A,96
     - A: REM ИЗТРИВАНЕ
 90  A = A + C
100  HCOLOR= 3
110  HPLOT 140 - A,96 - A TO 140 +
     A,96 - A TO 140 + A,96 + A TO
     140 - A,96 + A TO 140 - A,96
     -A: REM ЧЕРТАНЕ
115  REM *ИЗБОР НА СТРАНИЦА ЗА ИЗВЕЖДАНЕ*
120  POKE  - 16299 - D,0
130  D = 1 - D
140  NEXT J,I
150 B =  - B:C =  -C
160  GOTO 30
```

С ред 50 се избира графичната страница, в която се чертае. По време на чертането нейното съдържание не се извежда върху екрана. С редове 60 – 80 се изтрива намиращият се в страницата квадрат, като се начертава повторно с черен цвят (HCOLOR = 0). С редове 90 – 110 се чертае нов квадрат с по-голям или по-малък размер. След като в страницата се подготви новата фигура, с ред 120 се определя режимът, който я извежда на екрана.

Обърнете внимание, че в тази програма е реализирана компютърна анимация. Докато изображението на едната графична страница се извежда на екрана, в другата се чертае ново изображение, което, като замени предишното (чрез превключване на двете страници), създава впечатление за движещи се фигури.

ГЛАВА 2. РИСУВАНЕ В ГРАФИЧЕН РЕЖИМ HPC

Режимът HPC (режимът на груба графика) се използува често за рисуване, особено когато искаме да постигнем бързодействие, добри цветове, икономия на памет и прости програми. Има множество интересни игри, осъществени изцяло в режим HPC.

Върху екрана на микрокомпютъра се рисува, като се променя съдържанието на определени клетки от паметта. Затова е съществено да знаем съответствието между положението на петата върху екрана и адресите на паметта, които ги управляват. В началото на тази глава ще обясним това съответствие и ще опишем кодирането на цветовете в режим HPC. Ще разгледаме начините за рисуване на фигури и съхраняването им върху магнитен носител за следващо използуване. Накрая ще представим графичен редактор, който дава възможност да се рисуват и съхраняват картини в режим HPC.

2.1. ВРЪЗКА МЕЖДУ ТЕКСТОВА СТРАНИЦА И ТЕКСТОВ ЕКРАН

Да разгледаме действието на програма 2.1. При изпълнение тя запълва екрана с буквата Т, без да използува командата PRINT.

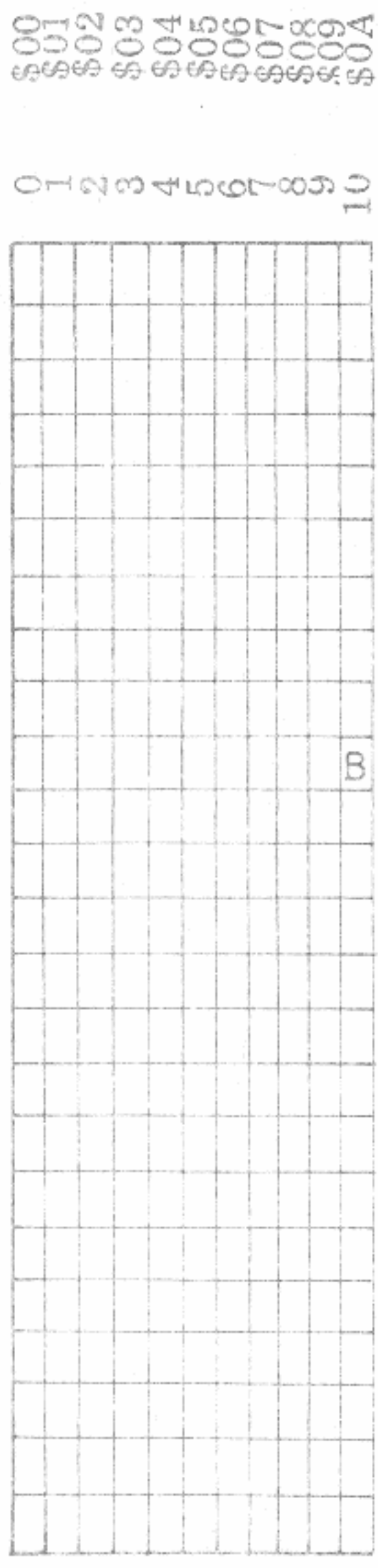
```
 1  REM **ПРОГРАМА 2.1**
 2  REM **ТЕКСТОВ ЕКРАН**
10  HOME
20  FOR I = 1024 TO 2047
30  POKE I,244
40  FOR J = 1 TO 50
50  NEXT J
60  NEXT I
70  PRINT  CHR$ (7)
80  FOR I = 1 TO 3000: NEXT I
90  GOTO 10
```

След като изчисти екрана с командата HOME (ред 10), програмата зарежда числото 244 в клетки с адреси от 1024 до 2047 ($400 – $7FF), т.е. в първата текстова страница. Числото 244 е кодът на буквата Т от кирилица. В резултат на това при изпълнение на програмата екранът се запълва изцяло с буквата Т. След кратка пауза, осъществена от цикъла на ред 80, управлението се предава в началото на програмата и тя се изпълнява отново. Ако в ред 30 запишем друг код вместо 244, буквата,която то се извежда, ще се промени.

При изпълнението на програмата ще забележите, че екранът се запълва на три части, въпреки че адресите от паметта се запълват последователно. Първият ред се появява в най-горната част на екрана. Вторият ред се разполага по-ниско, на около 1/3 от височината му, а третият — на 2/3. Четвъртият ред заема мястото непосредствено под първия, петият — под втория, а шестият — под третия. Това разместване при изписването на последователно следващите редове в трите области продължава до запълване на целия екран. То не е случайно и трябва да се има предвид от всеки, който програмира графични изображения. Въпреки че затруднява определянето на адресите, съответстващи на отделните полета от екрана, то се налага от електронната реализация на изображението и не може да се избегне.

Адресът на дадена позиция от екрана можете да определите, като използвате картата на паметта от фиг. 2.1. Първо определете мястото на знака, който искате да изведете върху екрана. След това намерете адреса, който му съответствува в паметта, като съберете числата, намиращи се на същия ред и колона, записани отляво и отгоре на картата. Например, ако искате да изведете знак на десета позиция от втори ред на екрана, трябва да съберете числата 1152 + 9 = 1161 ($480 + $09 = $489). Това е адресът, съответствуващ на желаната позиция. Обърнете внимание, че адресите над колоните са с единица по-малки от техните действителни номера.

$400	1024
$480	1152
$500	1280
$580	1408
$600	1536
$680	1664
$700	1792
$780	1920
$428	1064
$4A8	1192
$528	1320
$5A8	1448
$628	1576
$6A8	1704
$728	1832
$7A8	1960
$450	1104
$4D0	1232
$550	1360
$5D0	1488
$650	1616
$6D0	1744
$750	1872
$7D0	2000



фиг. 2.1

Използуването на картата на текстовата страница от паметта е демонстрирано в програма 2.2. В резултат от изпълнението на тази програма върху екрана се появява текстът АДРЕСАЦИЯ В ТЕКСТОВА СТРАНИЦА. Разположението му е показано на фиг. 2.1.

С ред 10 от програмата се изчиства екранът и се установява

```
 1  REM  **ПРОГРАМА 2.2**
 2  REM  **АДРЕСАЦИЯ В ТЕКСТОВА СТРАНИЦА**
10  HOME : TEXT
20  FOR AD = (1920 + 16) TO (1920 + 24)
30   READ KOD: POKE AD,KOD
40  NEXT AD
50  FOR AD = (1192 + 11) TO (1192 + 29)
60  READ KOD: POKE AD,KOD
70  NEXT AD
75  REM *КОДОВЕ ЗА ИЗВЕЖДАНЕ*
80  DATA 225, 228, 242, 229, 243, 225, 227, 233, 241
90 DATA  247, 160, 244, 229, 235, 243, 244, 239, 247, 225, 160,
    243, 244, 242, 225, 238, 233, 227, 225
```

текстов режим. С редове 20,30 и 40 се изписва първата част от текста (АДРЕСАЦИЯ). Кодовете за извеждане на съответните символи се четат от първия блок данни (ред 80) и се записват в адресите от паметта, съответствуващи на позициите от екрана, в които искаме да се появи надписът. С редовете 50, 60 и 70 се върши същото за втората част от текста (В ТЕКСТОВА СТРАНИЦА). Кодовете за извеждане на тази група символи се съдържат в ред 90.

Пълният набор от кодове за извеждане на символите в микрокомпютъра Правец – 82 е даден в табл. 2.1.

Таблица 2.1

Обратно извеждане				Мигащо извеждане				Нормално извеждане								
0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
$ 00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
0 0	@	P		0	@	P		0	@	P		0	Ю	P	@	П
1 1	A	Q	!	1	A	Q	!	1	A	Q	!	1	A	Q	A	Я
2 2	B	R	"	2	B	R	"	2	B	R	"	2	B	R	Б	Р
3 3	C	S	#	3	C	S	#	3	C	S	#	3	C	S	Ц	С
4 4	D	T	$	4	D	T	$	4	D	T	$	4	D	T	Д	Т
5 5	E	U	%	5	E	U	%	5	E	U	%	5	E	U	Е	У
6 6	F	V	&	6	F	V	&	6	F	V	&	6	F	V	Ф	Ж
7 7	G	W	'	7	G	W	'	7	G	W	'	7	G	W	Г	В

	Обратно извеждане					Мигащо извеждане					Нормално извеждане					
8	8	H	X	(	8	H	X	(	8	H	X	(	8	H	X	X	Ь
9	9	I	Y	)	9	I	Y	)	9	I	Y	)	9	I	Y	И	Ъ
10	A	J	Z	*	:	J	Z	*	:	J	Z	*	:	J	Z	Й	З
11	B	K	[	+	;	K	[	+	;	K	[	+	;	K	Ш	K	[
12	C	L	\	,	<	L	\	,	<	L	\	,	<	L	\	Л	Э
13	D	M	]	−	=	M	]	−	=	M	]	−	=	M	Щ	M	]
14	E	N	^	.	>	N	^	.	>	N	^	.	>	N	Ч	H	^
15	F	O	_	/	?	O	_	/	?	O	_	/	?	O	_	O	■

При изпълнение на програмата 2.1 ще забележите, че след всяко извеждане на символи в долната третина на екрана има малка пауза,преди извеждането да продължи в горната третина. Това се дължи на осем „губещи" се байта след всеки ред в долната третина. Тези байтове съществуват в паметта на микрокомпютъра, но не се извеждат върху екрана. За да се убедим в това, ще изчислим адреса на последната позиция в първия ред от долната третина на екрана (17 ред отгоре надолу): 1104 + + 39 = 1143. Следващият адрес, който се извежда след 1143,е адресът 1152, съответстващ на първата позиция от втория ред на горната третина. Между адрес 1143 и 1152 има осем клетки,които не се използуват при извеждане на изображението от първата страница в текстов режим и режим HPC. Тези „губещи" се клетки се използуват от операционната система на микрокомпютъра.

2.2. ВРЪЗКА МЕЖДУ ГРАФИЧНА СТРАНИЦА И ГРАФИЧЕН ЕКРАН

Поради това,че режимът HPC и текстовият режим използуват едни и същи области от паметта, те в много отношения си приличат. Например нека в програма 2.1 променим само ред 10, като вместо TEXT запишем GR.

Ако изпълним така променената програма, ще видим, че графичната част от екрана се изпълва с малки правоъгълни блокчета. Ако видеомониторът ни е цветен, ще забележим също, че те са подредени по двойки, състоящи се от разположени едно под друго тъмнозелено и бяло правоъгълно блокче. Четирите реда за текст в долната част на екрана се запълват отново със символа T. Ако добавим следния ред:

15 POKE −16302,0

с който установяваме режим на пълно графично изображение, целият екран ще се изпълни с двойки тъмнозелени и бели правоъгълничета.

Как се получават тези цветни правоъгълничета? За да отговорим, ще се върнем към програма 2.1. С ред 30 се записва числото 244($F4) във всяка клетка от първата страница на режим HPC. От друга страна, всеки байт от паметта в режим HPC съдържа информация, която определя цвета на две блокчета, разположени на екрана върху площта,която се заема от един символ в текстов режим. Информацията за символа и цвета на блокчетата е записана в една и съща клетка от паметта, но се възприема различно в зависимост от режима на извеждане. Например в текстовия режим числото 244 се дешифрира като знака T, а в режим HPC определя тъмнозелено и бяло правоъгълниче.

Кодовете на 16-те цвята, които могат да се запишат в половин байт (4 бита = тетрада), са:

0 ($0) черен	8 ($8) кафяв
1 ($1) пурпурен	9 ($9) оранжев
2 ($2) тъмносин	10 ($A) сив II
3 ($3) морав	11 ($B) розов
4 ($4) тъмнозелен	12 ($C) зелен
5 ($5) сив I	13 ($D) жълт
6 ($6) син	14 ($E) синьозелен
7 ($7) светлосин	15 ($F) бял

За да определим цветовете,които задава символът T в режим HPC, по-удобно е да разгледаме кода му в шестнадесетична система. Действително числото $F4(244) има $F в лявата и $4 в дясната тетрада на съответния байт. Съгласно таблицата за кодиране на цветовете тези стойности определят тъмнозелен цвят за горното и бял за долното правоъгълниче. Обърнете внимание,че дясната тетрада съдържа кода на цвета за горното, а лявата – за долното правоъгълниче. Използуването на тетради за кодиране на цвета на всяко блокче има известни неудобства, тъй като информацията в паметта може да се помества само побайтово. Това означава, че стойността, която се въвежда във всеки байт, трябва да се изчислява едновременно за двете блокчета, които той управлява.

След тези предварителни обяснения ние сме готови да рисуваме.

2.3. КАРТИНИ В РЕЖИМ HPC

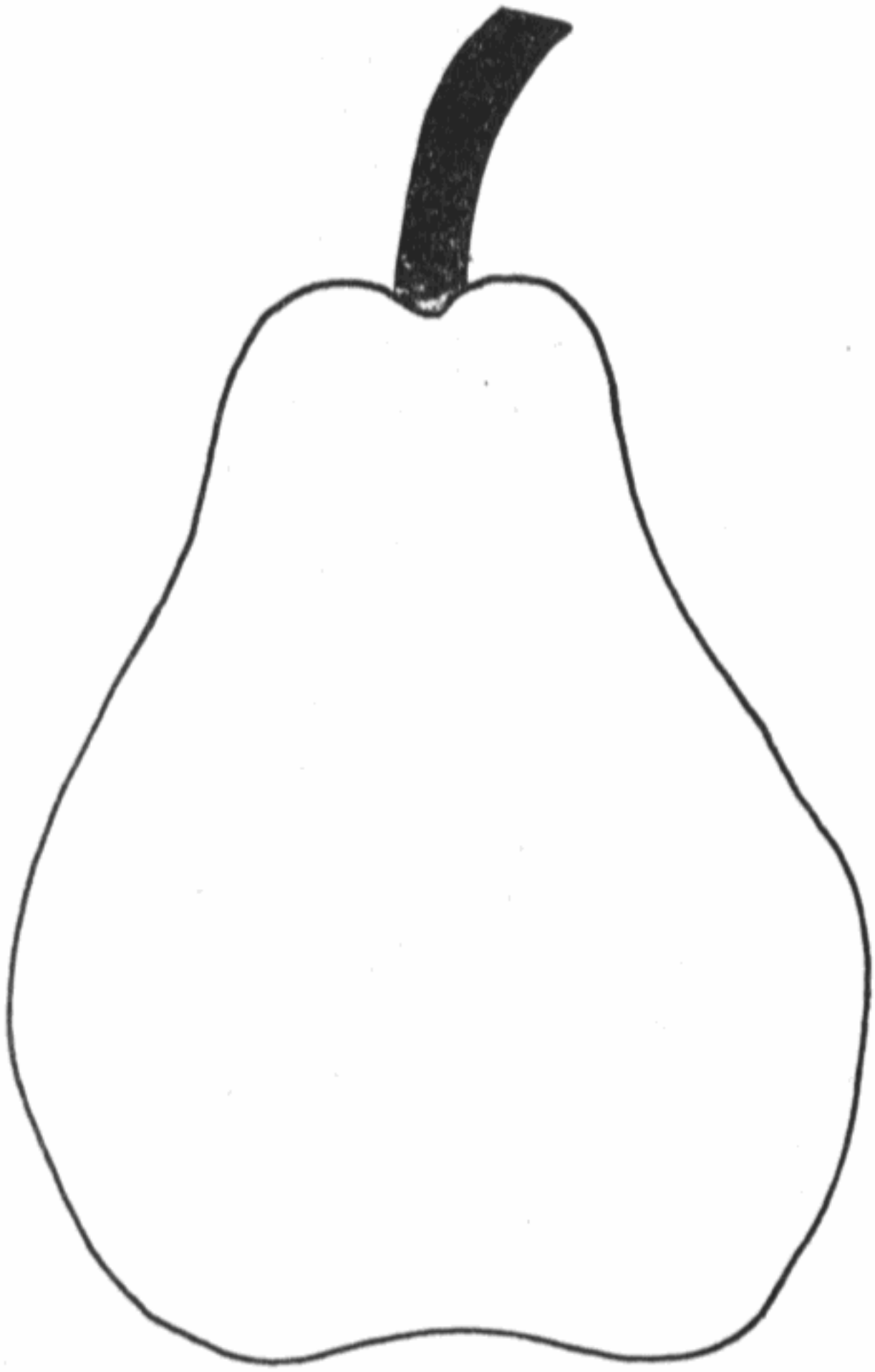
Създаването на картина или чертеж в режим HPC изисква добро предварително планиране и включва последователност от стъпки, които трябва да се изпълнят. За да демонстрираме този процес, ще преминем заедно през етапите, свързани с нарисуването на една круша.

На първата стъпка се създава предварителна скица на обекта, който искаме да изобразим на екрана. Скицата на нашата круша е показана на фиг. 2.2. На следващата стъпка трябва да запълним скицата с малките блокчета, които ни предоставя режим HPC. Това обаче не е толкова просто, колкото изглежда на пръв поглед. Усложнението се поражда от факта, че височината и широчината на блокчетата не са равни, а са в отношение приблизително 2:3. Ако не се вземе предвид това обстоятелство, ще се получат изкривявания във вертикалните и хоризонталните пропорции на рисунката. Най-простият начин за преодоляване на този проблем е да използуваме милиметрова хартия, която разчертаваме предварително на правоъгълничета с отношение на височината към широчината 2:3. След като подготвим милиметровата хартия, пречертаваме скицата на обекта върху нея и коригираме линиите ѝ в съответствие с възможностите, които ни дава режим HPC. Изображението на крушата, получено по този начин,е показано на фиг. 2.3.

Следващата стъпка е да определим кои блокчета от рисунката какъв цвят ще имат. Избраните цветове за крушата са отбелязани със съответните кодове на фиг. 2.3. След тази операция сме готови да изобразим нашата рисунка върху екрана на компютъра. Това можем да направим по два начина. Първият е да съставим програма, която използува командите PLOT,HLIN и VLIN, за да начертае фигурата. При втория начин можем да използуваме предварително написана програма — редактор, която ни дава възможност да нарисуваме крушата направо върху екрана и след това да съхраним изображението ѝ, като запишем информацията за него върху подходящ магнитен носител.

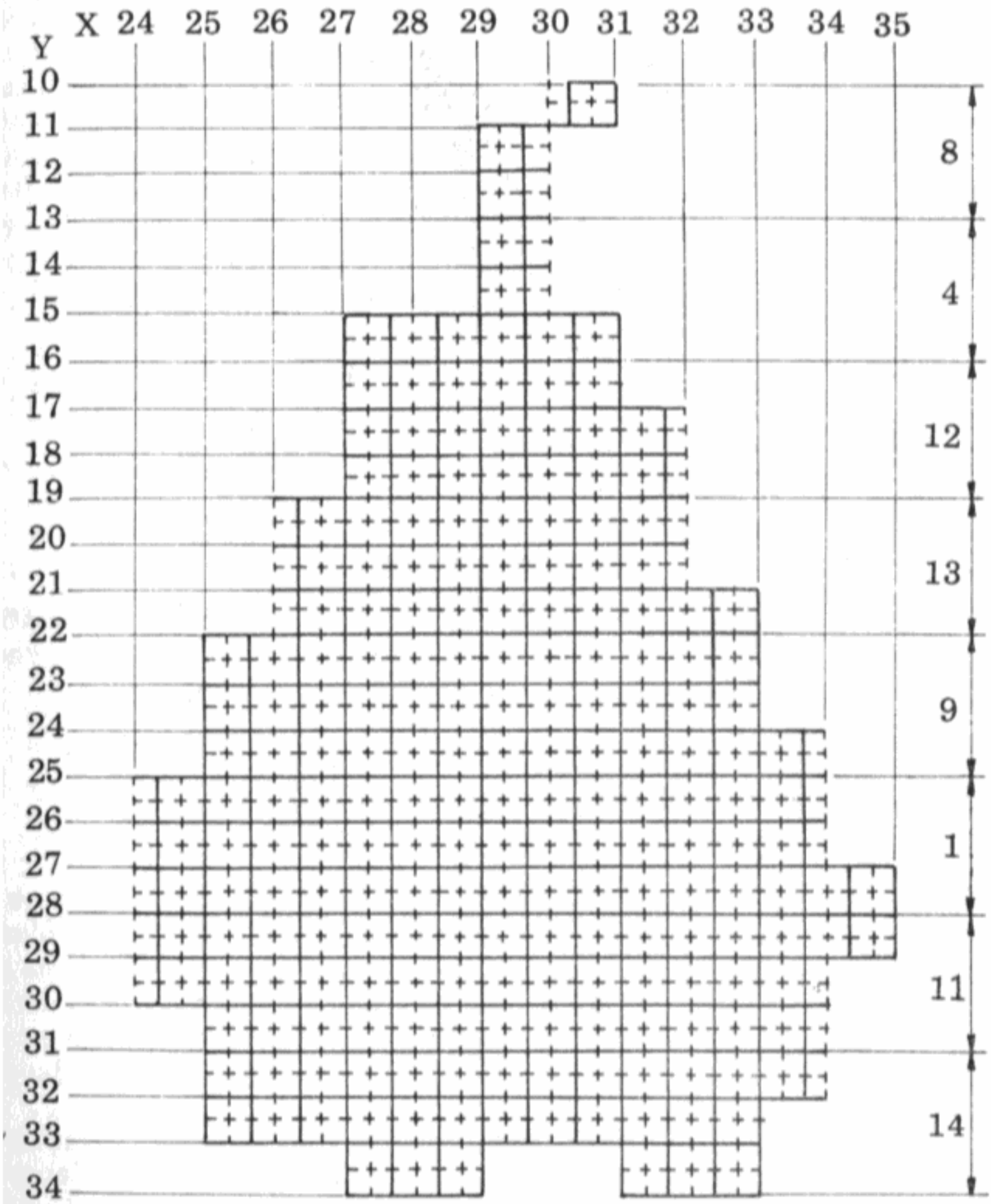
2.4. РИСУВАНЕ ЧРЕЗ СЪСТАВЯНЕ НА ПРОГРАМА

Най-простото решение на тази задача е да използуваме непосредствено командите PLOT, HLIN и VLIN. Това е направено в програма 2.3. На основата на скицата, която съставихме в предишната точка, тя чертае определен брой хоризонтални линии с отбелязаните цветове. Тъй като цветовете са разположени хоризонтално, използувана е главно командата HLIN. Употребата на една или друга команда при различните случаи зависи от особеностите на фигурата, която изобразяваме.



фиг. 2.2

```
 1  REM  **ПРОГРАМА 2.3**
 2  REM  **ЦВЕТНА РИСУНКА**
20  GR : REM НРС
40  COLOR= 8: REM КАФЯВО
50  PLOT 30,10
60  VLIN 11,12, AT 29
70  COLOR=4: REM ТЪМНОЗЕЛЕНО
80  VLIN 13,14 AT 29
90  HLIN 27,30 AT 15
100  COLOR= 12: REM ЗЕЛЕНО
```



фиг. 2.3

```
110  HLIN 27,30 AT 16
120  HLIN 27,31 AT 17
130  HLIN 27,31 AT 18
140  COLOR= 13: REM ЖЪЛТО
150  HLIN 26,31 AT 19
160  HLIN 26,31 AT 20
170  HLIN 26,32 AT 21
180  COLOR= 9: REM ОРАНЖЕВО
190  HLIN 25,32 AT 22
```

```
200  HLIN 25,32 AT 23
210  HLIN 25,33 AT 24
220  COLOR= 1: REM ПУРПУРНО
230  HLIN 24,33 AT 25
240  HLIN 24,33 AT 26
250  HLIN 24,34 AT 27
260  COLOR= 11: REM РОЗОВО
270  HLIN 24,34 AT 28
280  HLIN 24,33 AT 29
290  HLIN 25,33 AT 30
300  COLOR= 14: REM СИНЬО-ЗЕЛЕНО
310  HLIN 25,33 AT 31
320  HLIN 25,32 AT 32
330  HLIN 27,28 AT 33
340  HLIN 31,32 AT 33
350  END
```

Недостатъците на този метод се проявяват, когато трябва да се нарисува картина, състояща се от множество разноцветни блокчета и малко хоризонтални или вертикални линии.

2.5. РИСУВАНЕ ЧРЕЗ ЗАРЕЖДАНЕ НА ИНФОРМАЦИЯТА ЗА ИЗОБРАЖЕНИЕТО

Основната идея на този подход и неговите разновидности е да се начертае първоначално картината, а след това да се запази съдържанието на областта от паметта, в която е записана информацията за нея. По-нататък винаги когато се нуждаем от тази картина, ние можем да прехвърляме информацията за изображението от мястото, където сме я съхранили, в графичната страница, която управлява екрана.

Картината може да се изчертае или с програма, подобна на тази от предишната точка, или като се използува програма-редактор. Редакторите са готови програми, които ни облекчават при създаването, променянето и съхраняването на текст, графични изображения и музика. В зависимост от това те са текстови, графични или музикални. Графичните редактори от своя страна се делят на редактори за изображения в режим HPC и BPC.

Ще разгледаме графичен редактор за режим HPC със сравнително опростени възможности. Той е ядрото на редактор с разширени възможности, който се съдържа в приложената към книгата дискета. Редакторът, който се реализира от програмата 2.4 дава възможност да чертаем и да изтриваме отделни блокчета в режим HPC по целия екран без най-долните четири реда. След като стартираме програмата, трябва да зададем ко-

да на цвета, с който искаме да чертаем. След това мигащият маркер се появява в центъра на екрана. Ние можем да го движим нагоре, надолу, наляво и надясно, като натискаме съответно клавишите I, M, J и K. С клавишите U, O, N и < можем да движим маркера диагонално. Чрез клавиша D задаваме режим на чертане или придвижване без чертане. След като натиснем клавиша C, имаме възможност да променим цвета, с който чертаем. От редактора излизаме с клавиша ОСВ.

```
1  REM **ПРОГРАМА 2.4**
2  REM **НРС РЕДАКТОР**
10  HOME : GR :PF = 0
20 X = 20:Y = 20
30  GOSUB 230
35  REM *ПРОВЕРКА ЗА НАТИСНАТ КЛАВИШ*
40 C = PEEK ( - 16384): IF C > = 128 THEN C$ = CHR$ (C
- 128): POKE  - 16368,0: GOTO 60
50  PLOT X,Y: COLOR= 0: PLOT X,Y: COLOR= CN: GOTO 40
60  REM *АНАЛИЗ НА ВЪВЕДЕНИТЕ СИМВОЛИ*
70  IF PF THEN  PLOT X,Y
80  IF C$ = "J" AND X > 0 THEN X =  X - 1: GOTO 40
90  IF C$ = "I" AND Y > 0 THEN Y = Y - 1: GOTO 40
100  IF C$ = "K" AND X < 39 THEN X = X + 1: GOTO 40
110  IF C$ = "M" AND Y < 39 THEN Y = Y + 1: GOTO 40
120  IF C$ = "U" AND X > 0 AND Y > 0 THEN X = X - 1:Y =
     Y - 1: GOTO 40
130  IF C$ = "O" AND X< 39 AND Y > 0 THEN X = X + 1:Y
     = Y - 1: GOTO 40
140  IF C$ = "<" AND X< 39  AND Y < 39 THEN X = X + 1:Y
     = Y + 1: GOTO 40
150  IF C$ = "N" AND X > 0 AND Y < 39 THEN X = X - 1:Y
     = Y + 1: GOTO 40
160  IF C$ = "D" THEN PF = 1 - PF: GOTO 40
170  IF C$ = CHR$ (27) THEN GOTO 280
180  IF C$ = "C" THEN GOSUB 230
190  GOTO 40
200  REM *СМЯНА НА ЦВЕТА*
230  VTAB (24): INPUT „КОД ЗА НОВ ЦВЯТ (0-15)";CN$
240  CN =  VAL (CN$): IF CN < 0 OR CN > 15 THEN PRINT
     CHR$ (7): GOTO 230
250  PRINT : PRINT : PRINT : PRINT "КОД НА ЦВЯТ  = ";CN
260 COLOR= CN
270  RETURN
280  REM *ИЗЛИЗАНЕ ОТ РЕДАКТОРА*
300  PRINT "КРАЙ? (Д/Н)";: GET R$
310  IF R$ <  > "Д" GOTO 40
320  END
```

2.6. СЪХРАНЯВАНЕ НА КАРТИНА

Когато картината е готова върху екрана, има няколко начина да я съхраним за бъдещо използуване. Можем да я запишем върху магнитна лента чрез командата за запис върху лента на управляващата програма МОНИТОР. За целта първо извикваме програмата МОНИТОР и след това записваме съдържанието на първа страница ($400 – $7FF) с командите:

```
CALL -151
* 400.7FFW
```

Така записаното изображение можем да възстановим в същата област на паметта чрез командта за четене от магнитна лента:

```
* 400.7FFR
```

Ако разполагаме с дисково устройство, можем да съхраним картината чрез командите:

```
BSAVE КРУША,A$400,L$400
```

в случай на шестнадесетични числа и

```
BSAVE КРУША, A1024,L1024
```

в случай на десетични числа.

Изображението може да се възстанови на екрана с командата:

```
BLOAD КРУША
```

Параметрите A и L могат да се използуват, но не са задължителни.

Картината, която сме нарисували, можем да съхраним върху дискета и чрез команда от програма на БЕЙСИК, например:

```
15 PRINT CHR$(4) "BSAVE КРУША,A$400,L$400"
```

Информацията за изображение, което сме записали върху дискета, също можем да възстановим от програма на БЕЙСИК чрез командата:

```
25 PRINT CHR$(4) "BLOAD КРУША"
```

Ако запишете и възстановите изображението на екрана с която и да е от изброените команди, навярно ще забележите, че се появяват известни остатъци от предишното изображение в текстовата част на екрана. Това се получава, защото с командата BSAVE се записва целият екран, включително и текстът, намиращ се в най-долните четири реда. За да се избегне това, необходимо е, преди да я изпълните, да изчистите текстовата част от смесения екран.

Разгледаните дотук начини за съхраняване на нарисуваните картини в режим HPC са прости, но неикономични. За да се избегне записването на цялата страница ,когато информацията на картината заема незначителна част от нея, се прибягва до друг метод.

2.7. СКАНИРАНЕ НА ПАМЕТТА

При този подход паметта се претърсва байт по байт и се записва съдържанието и адресът само на онези байтове, чиято стойност е различна от нула. По този начин вместо цялата страница се съхраняват само байтовете, съдържащи информация за изображението. Ако имаме изображение не върху черен фон, а върху друг цвят, тогава е необходимо да се записват само байтовете, които имат стойност за цвят, различна от тази на фоновия цвят.

Паметта можем да сканираме по два основни начина. Първият от тях използува съответствието между позициите върху екрана и адресите от паметта, отразено в картата на първа страница за HPC от фиг. 2.1. В този случай се записва адресът на клетката, която съдържа байт, различен от фоновия цвят, и нейното съдържание. Вторият начин използува стандартната функция на БЕЙСИК SCRN (X,Y), която връща кода на цвета за позицията от екрана с координати X и Y. В този случай можем да запишем координатите на всяка позиция от екрана, чийто цвят се различава от фоновия цвят, заедно с кода на цвета ѝ.

```
 1  REM  **ПРОГРАМА 2.5**
 2  REM **ПРИМЕР1 – СКАНИРАНЕ НА ПАМЕТТА**
 5 BR = 0
10  FOR I = 1 TO 20
20  READ R
30  FOR S = R TO R + 39
40  IF  PEEK (S) <  > 0 THEN  PRINT S;" ,";PEEK (S):BR = BR
    + 1
50  NEXT S
60  NEXT I
70  PRINT "БРОЙ = ";BR;" КРАЙ"
80  REM *НАЧАЛНИ АДРЕСИ НА РЕДОВЕ*
90  DATA 1024,1152,1280,1408,1536
100  DATA 1664,1792,1920,1064,1192
110  DATA 1320,1448,1576,1704,1832
120  DATA 1960,1104,1232,1360,1488
```

С програмата 2.5 се претърсва паметта, в която е записана информация за изображението, и се извежда списък на всички адреси, съдържащи стойности, различни от нула, заедно с тези стойности. Съответствието между адресите от паметта и началните позиции за всеки ред на екрана е отразено в блока данни, записан в последните четири реда на програмата (90 – 120). Адресите за един ред се сканират чрез цикъла, реализиран в редове 30 до 50. По този начин екранът се претърсва отгоре надолу, като се избягва проверката на клетките от първа текс-

това страница, които нямат съответстващи позиции върху екрана. С ред 40 се извежда адресът и съдържанието на всеки байт, различен от нула, и се увеличава с единица броячът на ненулевите байтове. След като се претърси цялата страница, с ред 70 се отпечатва броят на ненулевите байтове и се извежда съобщение за край на изпълнението.

```
 1  REM **ПРОГРАМА 2.6**
 2  REM **ПРИМЕР2 – СКАНИРАНЕ НА ПАМЕТТА**
10  FOR Y = 1 TO 39
20  FOR X = 1 TO 39
30 T = SCRN( X,Y): IF T <  > 0 THEN PRINT X;" ,";Y;",";T:BR =
    BR + 1
40  NEXT X: NEXT Y
50  PRINT "БРОЙ = ";BR;" КРАЙ"
60  END
```

Програмата 2.6 демонстрира втория начин за сканиране на паметта. Тя използува стандартната функция на БЕЙСИК SCRN (X,Y). Координатите X и Y се променят от 0 до 39 (редове 10 – 40) и се проверява кодът на цвета за всяка позиция. Ако той е различен от нула, върху екрана се извеждат координатите X,Y и стойността му. С ред 50 се извеждат броят на ненулевите позиции и съобщение за край.

За да пробвате действието на програмите 2.5 и 2.6, необходимо е да се намирате в режим HPC и да имате някаква картина върху екрана. Ако сте създали файла КРУША от предишната точка, сега е необходимо да въведете една от описаните програми и да зададете последователно командите:

```
GR
BLOAD КРУША
RUN
```

В резултат трябва да получите списък от адреси на клетки и техните съдържания. Поради това, че в момента на извеждане на информацията микрокомпютърът е в режим на смесен екран, числата се помещават в долните четири реда на екрана. Те се движат бързо и ако са повече от четири реда, трудно могат да се запишат. Затова по-удобно е да ги извеждаме на печатащо устройство. Ако то е свързано към куплунг номер 1, необходимо е да вмъкнем в двете програми следния нов ред:

```
8  PR#1
```

Тогава информацията за изображението на екрана ще се извежда на печатащо устройство.

Списъците, съдържащи информация за предварително създаденото изображение, можем да използуваме в нови програми като блокове от данни, които да четем с командата READ и да зареждаме чрез командата POKE в паметта на компютъра. Този

подход има предимството, че не използува дисково устройство, но е трудно приложим, тъй като изисква ръчно въвеждане на данните. Това е трудопоглъщаща работа, свързана с допускането на грешки. Поради това ще разгледаме как се съхранява информацията за изображението в текстов файл.

2.8. ТЕКСТОВИ ФАЙЛОВЕ

Вместо да извежда списък на адреси и техните съдържания, програмата 2.5 може да се преобразува по такъв начин, че да съхранява информацията за изображението върху дискета. Нейната модификация (програма 2.7) създава файл, съдържащ същия списък от адреси и техните съдържания, който се извежда от програмата 2.5.

```
 1  REM **ПРОГРАМА 2.7**
 2  REM **СЪЗДАВАНЕ НА ТЕКСТОВ ФАЙЛ**
 4  D$ =  CHR$ (13) +  CHR$ (4)
 5  INPUT "ИМЕ НА ФАЙЛ:";IME$
 6  PRINT  D$"OPEN"IME$
 7  PRINT  D$"DELETE"IME$
 8  PRINT  D$"OPEN,"IME$
 9  PRINT  D$"WRITE"IME$
10  FOR I = 1 TO 20
20  READ R
30  FOR S = R TO R + 39
40  IF  PEEK (S) <  > 0 THEN  PRINT S: PRINT  PEEK (S)
50  NEXT S
60  NEXT I
70  PRINT ":*"
75  PRINT D$"CLOSE"IME$
80  PRINT  CHR$ (7)
85  END
90  DATA 1024,1152,1280,1408,1536
100  DATA 1664,1792,1920,1064,1192
110  DATA 1320,1448,1576,1704,1832
120  DATA 1960,1104,1232,1360,1488
```

С ред 5 се задава името на файла, в който искаме да съхраним данните, а с ред 70 се отбелязва краят му със знака *. За да пробвате програмата, трябва да я въведете и да зададете отново командите:

```
GR
BLOAD КРУША
RUN
```

Когато на екрана се появи текстът ИМЕ НА ФАЙЛ, нека въ-

ведем новото име КРУША.1. Тогава ще започне сканиране на първа страница за режим HPC и информацията за изображението ще се съхрани върху дискетата. Краят на работата на програмата се отбелязва с кратък звуков сигнал (ред 80).

Данните, записани по този начин, могат да се прочетат от файла КРУША.1 и да се поместят в паметта чрез програма 2.8. Тя е оформена като подпрограма.

```
2001  REM **ПРОГРАМА 2.8**
2002  REM **ИЗОБРАЖЕНИЕ ОТ ТЕКСТОВ ФАЙЛ**
2010  GR
2020  D$ = CHR$ (13) + CHR$ (4)
2030  IME$ = "КРУША.1"
2040  PRINT D$"OPEN"IME$
2050  PRINT D$"READ"IME$
2060  INPUT ADR$
2070  IF ADR$ = "*" GOTO 2120
2080  ADR = VAL (ADR$)
2090  INPUT KOD
2100  POKE ADR,KOD
2110  GOTO 2060
2120  PRINT D$"CLOSE"IME$
2130  RETURN
```

За да се стартира програмата 2.8, необходимо е да се извика от друга програма, например:

```
10  REM **СТАРТИРАНЕ НА ПОДПРОГРАМА**
20  GOSUB 2001
30  END
```

След като програмата 2.8 се въведе в компютъра и се стартира извикващата я програма, екранът се изчиства и от дискетата се зарежда информацията за изображението, което сме съхранили в зададения файл.

Този метод за съхраняване на картини спестява неудобството от ръчното въвеждане на големи масиви от данни, но достъпът до дискетата е сравнително бавен. Това може да забави изпълнението на програмата, използуваща предварително нарисувани картини. Този недостатък се избягва чрез методите, описани в следващите точки.

2.9. ИЗПЪЛНИМИ ФАЙЛОВЕ

Идеята на този подход е да се създадат текстови файлове, съдържащи команди DATA, последвани от двойки стойности, които представляват съответно адреси на клетки от страницата на изображението и техните съдържания. Записите на тези файлове се формират като номерирани редове от програма на

БЕЙСИК, които могат да се добавят и изпълняват с програми, писани впоследствие. С програма 2.9 се създават изпълними файлове. Тя генерира по десет двойки стойности след всяка команда DATA. За последната команда DATA те могат да са по-малко. Редовете, съдържащи блоковете с данни, започват от номер 2000 и номерата им се увеличават през 5. С ред 4 задаваме името на файла, който искаме да създадем, а в ред 9 се установява номерът на първия ред от блока данни да бъде равен на 2000.

```
 1  REM **ПРОГРАМА 2.9**
 2  REM **СЪЗДАВАНЕ НА ИЗПЪЛНИМ ФАЙЛ**
 3  D$ =  CHR$ (13) +  CHR$ (4)
 4  INPUT "ИМЕ НА ФАЙЛ:";IME$
 5  PRINT D$"OPEN"IME$
 6  PRINT D$"DELETE"IME$
 7  PRINT D$"OPEN"IME$
 8  PRINT D$"WRITE"IME$
 9  NR% = 2000:B = 1:BR = 0
10  FOR I = 1 TO 20
20  READ R
30  FOR S = R TO R + 39
40  IF  PEEK (S) = 0 GOTO 50
42  BR = BR + 1
44  IF B = 1 THEN  PRINT NR%;"DATA";
46  IF B > 1 THEN  PRINT ",";
47  PRINT S;" ,"; PEEK(S);:B = B + 1
48  IF B > 10 THEN B = 1:NR% = NR% + 5: PRINT
50  NEXT S
60  NEXT I
65  PRINT D$"CLOSE"IME$
70  PRINT "БРОЙ = ";BR;" КРАЙ";CHR$(7)
80  REM *НАЧАЛНИ АДРЕСИ НА РЕДОВЕ*
90  DATA 1024,1152,1280,1408,1536
100  DATA 1664,1792,1929,1064,1192
110  DATA 1320,1448,1576,1704,1832
120  DATA 1960,1104,1232,1360,1488
```

Сега можем да пробваме програмата по същия начин, както направихме това за програмата, създаваща текстов файл. За целта въвеждаме последователно в микрокомпютъра командите:

```
GR
BLOAD КРУША
RUN
```

Веднага след стартирането трябва да зададем името на файла, който искаме да създадем. Тъй като той трябва да се

различава от предишните файлове, ще отговорим с новото име КРУША.2. След това дискетата се завърта и изпълнимият файл се записва върху нея. В края на изпълнението на програмата се издава звуков сигнал и се извежда информация за броя на записаните двойки байтове.

По такъв начин в новия файл КРУША.2 имаме готови редове от програма на БЕЙСИК. Те се състоят от номера на редове и команди DATA, последвани от по десет двойки стойности, представляващи информация за изображението, което сме създали предварително. Тези команди DATA лесно могат да бъдат долепени към всяка нова програма. За да демонстрираме това, нека въведем в микрокомпютъра следната последователност от команди:

```
NEW
10 GR
20  BROI=XXX
30 FOR I = 1 TO BROI
40 READ ADR,KOD:POKE ADR,KOD
50 NEXT I
60 END
EXEC КРУША.2
```

Стойността,която се присвоява на променливата BROI в ред 20, трябва да е равна на броя на двойките стойности (адрес, съдържание), записани в изпълнимия файл. Тази информация се извежда винаги в края на изпълнението на програмата, която е създала успешно изпълним файл и трябва да се отбелязва за бъдещо използуване. Разбира се, ако усложним програмата 2.9, тази информация може да се записва автоматично в самия файл.

След изпълнението на командата EXEC КРУША.2 дискетата се завърта и когато спре, ако въведем командата LIST, ще видим редове от 10 до 60 от програмата на БЕЙСИК заедно с долепени към тях редове, съдържащи команди DATA и започващи от номер 2000. Ако стартираме така оформилата се програма с команда RUN, на екрана ще се изчертае изображението, съхранено от нас в изпълнимия файл.

Тъй като блокът от данни, съдържащ информация за изображението, сега е част от програмата на БЕЙСИК, тя не се нуждае от достъп към дискетата, за да зарежда информацията за изображението през време на изпълнението си. По този начин се преодоляват недостатъците, свързани с бавния достъп до дискетата и ръчното въвеждане на блока от данни.

2.10. ТРЕТА СТРАНИЦА ЗА РЕЖИМ HPC

Досега разполагахме информацията за изображението върху външен магнитен носител или в самата програма. Съществува

и трета възможност, при която можем да съхраним информацията в паметта на микрокомпютъра. Отбелязахме, че втората страница на режим HPC трудно може да се използува за тази цел. За това областта, която ще използуваме временно за съхраняване на изображението, можем да наречем трета страница. Разполагането на информацията в определена област от паметта ни позволява бързо да я прехвърлим в първа страница и по този начин да я изведем на екрана. Това може да се осъществи с програми на БЕЙСИК или на машинен език.

Първото нещо, което трябва да направим в този случай, е да заредим информацията за изображението в защитена и неизползувана за други нужди област от паметта на микрокомпютъра. Ние ще използуваме старшите $400 байта от свободната памет, предоставена за нуждите на потребителя. Те се намират между адреси $9200 – $95FF (37376 – 38399). Информацията за изображението можем да заредим в тази област чрез командите:

BLOAD КРУША,A$9200 или
BLOAD КРУША,A37376

Ако изображението е съхранено върху магнитна лента,трябва да използуваме командата на МОНИТОРА:

*9200.95FFR

След като запишем информацията за изображението в желаната област от паметта, необходимо е да я защитим от разрушаване. За целта променяме съдържанието на указателя, сочещ горната граница на свободната памет, достъпна за програмите на БЕЙСИК, така че да стане равно на първия адрес, намиращ се под областта, в която сме разположили нашата трета страница. Това постигаме чрез:

HIMEM: 37375

Сега можем да прехвърлим информацията за изображението от трета в първа страница чрез командата на МОНИТОРА:

*400<9200.95FFM

В резултат изображението, което сме съхранили предварително, трябва да се появи на екрана.

Описаната последователност от команди можем да изпълним наведнъж, ако обединим всички операции в обща програма.

```
 1  REM **ПРОГРАМА 2.10**
 2  REM **ПРЕМЕСТВАНЕ НА ИЗОБРАЖЕНИЕ В ПАМЕТТА**
10  HIMEM: 37375
20 D$ =  CHR$ (13) +  CHR$ (4)
30  PRINT D$;"BLOAD КРУША,A37376"
40   GR
50  FOR I = 0 TO 1023
60  POKE 1024 + I, PEEK (37376 + I)
70  NEXT I
80  END
```

С ред 10 се установява горната граница на паметта, достъпна за програмите на БЕЙСИК, под най-малкия адрес, който ще използуваме за трета страница. С ред 30 информацията за изображението се зарежда от дискетата в защитената по този начин област от паметта на микрокомпютъра. Цикълът в редове от 50 до 70 прехвърля байт по байт информацията от областта с адреси 37376 – 38399 (37376+1023) в първата страница на режим HPC. В резултат от изпълнението на програмата изображението, което сме записали върху дискетата, се появява на екрана. Недостатък на тази програма е сравнително бавното прехвърляне на информацията от една област на паметта в друга. Този процес се ускорява значително, ако вместо на БЕЙСИК цикълът за прехвърлянето на информацията се реализира на машинен език.

Такъв цикъл можем да вмъкнем в програма на БЕЙСИК с последователност от команди POKE или блок от данни, записан след команди DATA. Ние ще използуваме първия начин, като в програмата включим командите:

```
POKE 768,160:POKE 769,0
POKE 770,76:POKE 771,44
POKE 772,254
```

Така въведената програма на машинен език ще стартираме с командата CALL 768. Преди това е необходимо да зададем:

начален адрес на областта, източник на данните;

краен адрес на областта, източник на данните;

начален адрес на областта, получател на данните.

Всеки от тези три адреса ще зададем в по два байта от паметта по следния начин:

в адрес 60 – младшия байт на началния адрес на областта-източник;

в адрес 61 – старшия байт на началния адрес на областта-източник;

в адрес 62 – младшия байт на крайния адрес на областта-източник;

в адрес 63 – старшия байт на крайния адрес на областта-източник;

в адрес 66 – младшия байт на началния адрес на областта-получател;

в адрес 67 – старшия байт на началния адрес на областта-получател.

```
 1  REM **ПРОГРАМА 2.11**
 2  REM **ПРЕМЕСТВАНЕ НА ДАННИ В ПАМЕТТА ЧРЕЗ МАШИННА ПРОГРАМА**
10  HIMEM: 37375
20 D$ =  CHR$ (13) +  CHR$ (4)
30  PRINT D$;"BLOAD КРУША,A37376"
```

```
40  REM *МАШИННА ПРОГРАМА*
50  POKE 768,160: POKE 769,0
60  POKE 770,76: POKE 771,44
70  POKE 772,254
80  REM *АДРЕСИ НА ОБЛАСТИ*
90  POKE 60,00: POKE 61,146
100  POKE 62,255: POKE 63,149
110  POKE 66,0: POKE 67,4
120  GR
130  CALL 768
140  END
```

Програмата 2.11 включва предложената програма на машинен език (редове 50 – 70). Адресите, определящи областите източник и получател на информацията, са указани с трите двойки команди POKE в редове 90 – 110. След това машинната програма се стартира с ред 130. Изображението се появява върху екрана много по-бързо, отколкото при изпълнението на програма 2.10. Машинната програма е удобна за използуване, защото може да се зареди предварително в паметта на микрокомпютъра заедно с една или повече картини и след това да се стартира за различни адреси на областите източник и получател на информацията.

Подготовката на десетичните адреси, определящи тези области, се извършва по следния начин. Взема се целият шестнадесетичен адрес (например $9200). Разделя се на две части ($92 и $00) и се изчислява десетичната стойност, която трябва да се постави във всяка двойка байтове, определящи адрес. Например за $92 изчисляваме 9x16 + 2 = 146, а за $00 = 0. Важно е така получените двойки стойности да запишем в съответните двойки байтове. Младшата част на адреса се поставя винаги в първия, а старшата – във втория байт.

Дотук разгледахме основните методи за създаване и съхраняване на изображения в режим HPC. Всеки от тях трябва да се използува съобразно условията и нуждите, които трябва да се удовлетворяват. За онези, които ще използуват в програмите си предварително създадени изображения, съхранени по подходящ начин, е предложен графичен редактор за режим HPC.

2.11. ГРАФИЧЕН РЕДАКТОР ЗА РЕЖИМ HPC

Описаната програма-редактор се съдържа в приложената към книгата дискета и обединява разгледаните методи за построяване и съхраняване на изображения в режим HPC. Редакторът дава възможност да се рисува непосредствено върху екрана на видеомонитора, да се съхраняват изображенията върху магни-

тен носител и да се възстановяват по някой от описаните начини. Програмата е записана върху дискетата с името ПР 2.12.

Основните функции, които осъществява редакторът, се изписват върху екрана веднага след стартиране на програмата. Те съставят главното меню на редактора, което има вида:

1 – редактиране;
2 – запис;
3 – четене(зареждане);
4 – край.

Като въведем цифра от 1 до 4, избираме някоя от функциите на редактора и управлението се предава на тази част от програмата, която я осъществява. Например, ако изберем 1, влизаме в режим

1 – редактиране

При него екранът се разделя на две части – текстова и графична (смесен режим). Горните 20 реда са предоставени за чертане, а долните 4 реда се използват за обмен на съобщения между микрокомпютъра и потребителя. Веднага след преминаване в режим на редактиране в лявата част от текстовата област на екрана се изписва съобщението ЧЕРТАЕ, което означава, че при движението си маркерът ще оставя следа върху екрана, т.е. ще чертае. В обратния режим „не чертае" се преминава, като се натисне клавишът Д/D (независимо дали сме в режим на кирилица или латиница). В този случай движението на маркера не оставя следа върху екрана. Това състояние се отбелязва в текстовата област със съобщението НЕ ЧЕРТАЕ. Вдясно от него се появява надпис ЦВЯТ:15. Той показва, че в режим на чертане ще се използува цвят 15, т.е. бял. Ако искаме да чертаем с друг цвят, необходимо е да натиснем клавиша Ц/С. Тогава след надписа ЦВЯТ: се появява знакът „?". Сега сме задължени да въведем номера на новия цвят. Нека натиснем 8 и клавиша RETURN. Надписът върху екрана сега е ЦВЯТ:8 и означава, че от този момент нататък ще чертаем с цвят 8 (кафяв), докато сменим отново цвета с клавиш Ц/С.

Движението на маркера в графичната част от екрана се управлява чрез осем клавиша. Всеки от тях може да се използува както в режим кирилица, така и в режим латиница. Клавишът И/I движи маркера нагоре, М/M – надолу, К/K – надясно и Й/J – наляво. Останалите четири клавиша придвижват маркера по диагонал. Клавишът У/U го премества нагоре и наляво, О/O – нагоре и надясно, ,/< – надолу и надясно и Н/N – надолу и наляво.

От режим на редактиране се излиза, като се натисне клавишът ОСВ. На екрана се появява отново главното меню. От него, ако въведем цифрата 2, ще преминем в режим

2 – запис.

При този режим се извежда следното меню:
1 – запис в двоичен файл;
2 – запис в текстов файл;
3 – запис в изпълним файл;
4 – връщане към главното меню.

С първата функция се записва съдържанието на цялата текстова страница (клетки с адреси 1024 – 2047) върху дискета чрез командата BSAVE.

С втората функция се сканира екранът и се отбелязват координатите и кодовете на цветовете на онези точки от екрана, чийто цвят се различава от фоновия. Тези данни се записват върху дискетата като текстов файл. Сканирането на екрана е сравнително бавен процес, по време на който няма видим резултат от изпълнението на програмата. Затова след сканирането на всеки ред се чува кратък сигнал, по който се съди за продължаващото изпълнение на програмата.

С третата функция се създава файл, който след изпълнение с командата EXEC оставя програма в паметта. Като изпълним тази програма, можем да възпроизведем фигурата, която се е намирала на екрана по време на изпълнение на функцията. При създаването и на този файл се сканира екранът, което е съпроводено със звукова индикация.

Четвъртата функция ни връща към главното меню. След като въведем цифрата 3, редакторът преминава в режим

3 – четене(зареждане).

При този режим се извежда меню с три функции:
1 – четене на двоичен файл;
2 – четене на текстов файл;
3 – връщане към главното меню.

При първата функция се чете двоичен файл от дискета и се зарежда в първата текстова страница.

При втората функция се чете текстов файл, който съдържа информация за координатите и цветовете на онези точки от екрана, които осъществяват изображението. Тази информация активизира съответните адреси от първа текстова страница и по този начин възстановява съхранената върху дискета картина.

И в двата случая се осъществява диалог, за да се определи името на файла, от който се зарежда информацията.

Третата функция ни връща към главното меню на редактора. От него можем да излезем окончателно, като изберем цифрата 4 – край.

ГЛАВА 3. **РИСУВАНЕ В ГРАФИЧЕН РЕЖИМ ВРС**

Точността на изображението върху екрана на микрокомпютъра зависи от разделителната му способност. Последната се определя от броя на точките върху единица площ, чиято яркост и цвят можем да управляваме. В това отношение графичният режим с висока разделителна способност има съществени предимства пред режима НРС. Той позволява да се управлява всяка от 53 760-те елементарни точки от изображението на екрана, които са двадесет и осем пъти по-малки от правоъгълничетата, използувани в режим НРС. В резултат на това получените изображения са значително по-добри, но за сметка на известни усложнения и загуби. Адресирането на паметта, управляваща точките на изображението, се усложнява, а обемът ѝ се увеличава четири пъти. Към това трябва да се добавят намаленият брой цветове и невъзможността да ги поставяме произволно във всяка точка от екрана. Съществуват множество графични редактори, предназначени за създаване на изображения в режим ВРС. Тяхното използуване дава значителни удобства и облекчава рисуването. Въпреки това необходимо е да се знаят основните начини за програмиране на изображения върху екрана и управлението на яркостта и цветовете на отделните му точки.

3.1. УПРАВЛЕНИЕ НА ЕКРАНА

Има две области от паметта, които се използуват за съхраняване на информацията за изображението в режим ВРС. Едната се нарича *първа графична страница за режим ВРС* и е разположена от адрес 8192 до адрес 16383 ($2000 – $3FFF). Другата се нарича *втора графична страница за режим ВРС* и е между адреси 16384 и 24575 ($4000 – $5FFF). Почти всички примери, дадени в тази глава, работят с първата графична страница. Демонстрираните похвати и програми могат да се използуват по същия начин за втората графична страница. Необходимо е само да се добави числото 8192 ($2000) към съответния адрес от първа страница.

За да обясним как се разполага в паметта на микрокомпютъра информацията, управляваща всяка точка от екрана в режим ВРС, ще използуваме програма 3.1.

```
1  REM **ПРОГРАМА 3.1**
2  REM **УПРАВЛЕНИЕ НА ТОЧКИТЕ ОТ ЕКРАНА
   В РЕЖИМ ВРС**
```

```
10   HGR
20   POKE - 16302,0
30   REM *ЦИКЪЛ*
40  FOR I = 8192 TO 16383
50  FOR J = 1 TO 8
60  A = 2 ^ ( 8- J)
70   POKE I,A
80   NEXT J
90   NEXT I
100 PRINT CHR$ (7)
110 END
```

Тя установява последователно в единица един след друг всички битове на клетките от паметта в първата графична страница. Тъй като командите и функциите на БЕЙСИК нямат достъп до отделни битове от паметта, последните се превключват чрез презаписване на целия съдържащ ги байт. Стойностите, необходими за последователно установяване в единица на всички битове от един байт, се генерират със следния цикъл:

```
FOR J = 1 TO 8
A = 2 ^ (8 - J)
NEXT J
```

Преместването на „вдигнатия" бит в рамките на един байт в резултат от изпълнението на този цикъл е показано в табл. 3.1. При изпълнение на програмата 3.1 върху екрана се получават резултати, които първоначално изглеждат нелогични. На тяхното обяснение ще обърнем по-специално внимание.

Таблица 3.1

J	$2^{(8-J)}$	A	Съдържание на байта
1	2^7	128	1000 0000
2	2^6	64	0100 0000
3	2^5	32	0010 0000
4	2^4	16	0001 0000
5	2^3	8	0000 1000
6	2^2	4	0000 0100
7	2^1	2	0000 0010
8	2^0	1	0000 0001

След стартирането на програмата се появява светеща точка близо до горния ляв ъгъл на екрана. Тя започва да се движи бързо наляво. След седем стъпки спира. В този момент вдясно от нея се появява втора светеща точка, която след седем стъпки наляво спира точно там, където се появи първата. Този процес се повтаря, докато целият ред се запълни със светещи точ-

ки, отстоящи на седем стъпки една от друга. След него по същия начин се запълва следващ ред и процесът продължава, докато целият екран се запълни с 40 колони подредени една под друга точки.

Несъответствуващ на очакванията ни е фактът, че битовете, които установяваме в единица, се преместват отляво надясно, а светещата точка се движи отдясно наляво.Това се получава, защото микрокомпютърът Правец-82 извежда битовете на всеки байт в обратен ред върху екрана, т.е. най-десният бит на байта се разполага в най-левия край на съответното поле, а най-левият бит в най-десния край.

Изисква допълнително обяснение фактът, че когато местим установения в единица бит в рамките на байта, светещата точка на екрана изминава само седем стъпки. Това е така, защото информацията от осмия (най-левия) бит на всеки байт не се извежда непосредствено и той няма съответствуваща му точка върху екрана. Предназначението на този бит е да съхранява информация за цвета на останалите седем бита на байта. Как се използува той, ще разберем по-нататък.

В режим HPC наблюдавахме непоследователно запълване на разположените един до друг редове от екрана. И сега екранът се разделя на три части (секции). Най-напред се изчертава един ред точки в горната му част, след това – в средната, а най-накрая – в долната. След като се запълни ред от долната секция, следва пауза, преди да започне да се запълва ред от най-горната. Така процесът се повтаря. Новото, което се наблюдава при този режим е, че даже редовете вътре в една секция не се запълват последователно. След като в секцията е изчертан един ред, следващият запълван ред е на седем стъпки по-ниско. Непосредствено под първия ред започва да се чертае нов ред едва след като цялата секция се запълни с редове, отстоящи на седем стъпки един от друг.

3.2. КАРТА НА ПАМЕТТА В РЕЖИМ ВРС

Дотук стана ясно, че на всяка комбинация битове в даден байт от паметта съответствува определена комбинация от светещи точки върху екрана. Тъй като се извеждат седем бита от един байт, съществуват 128 възможни комбинации. Няма да разглеждаме всичките, но ще насочим вниманието ви върху следните:

10000000	НННННН
01000000	НННННС
00100000	ННННСН
00010000	НННСНН

```
00001000        HHHCHHH
00000100        HHCHHHH
00000010        HCHHHHH
00000001        CHHHHHH
```

Те показват коя точка от полето, управлявано от даден байт, свети, когато един бит от него е установен в единица. Светещите точки са означени със С, а несветещите – с Н. Отново се вижда, че за най-левия бит на всеки байт няма съответствуваща светеща точка.

За да управляваме определени точки от екрана, освен съответствието между битове и точки необходимо е да знаем и адреса на байта, съдържащ съответния бит. Адресирането на байтовете, съдържащи информацията за изображението в режим BPC, ще обясним чрез картата на паметта, дадена на фиг. 3.1 *а*.

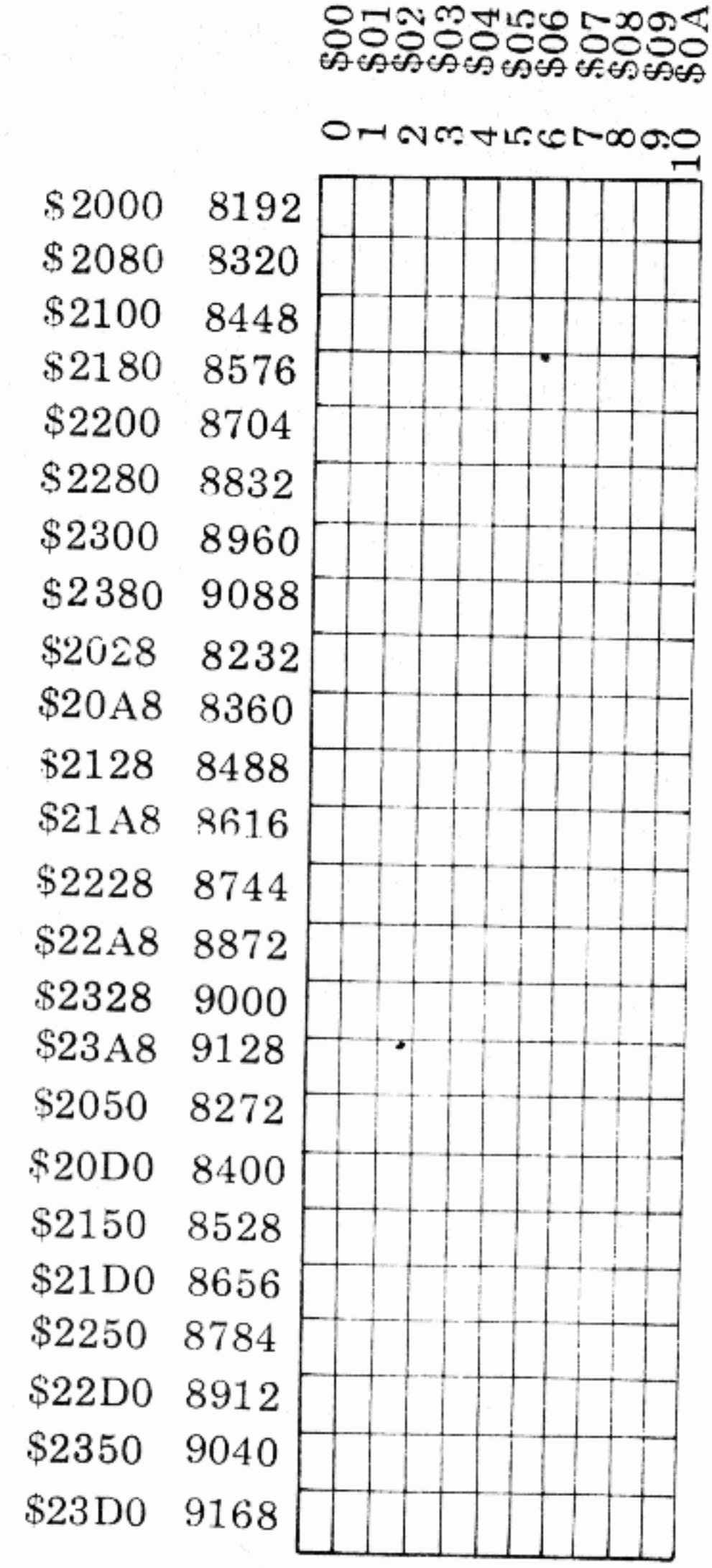
Показаните в картата правоъгълничета до голяма степен приличат на тези от режим HPC. Адресът на всяко от тях изчисляваме, като сумираме числата, стоящи срещу реда и колоната, на които то се намира. Едно правоъгълниче се състои от осем реда, съдържащи по седем точки. Всеки от тези редове се управлява от отделен байт. Адресът на този байт изчисляваме, като към адреса на правоъгълничето добавим произведението от номера на реда, който той управлява, и 1024. Редовете в правоъгълничето се номерират отгоре надолу, като се започва от нула (фиг. 3.1.б).

Следователно, за да управляваме определена точка от екрана, първо трябва да определим в кое правоъгълниче се намира. След това трябва да определим номера на реда в правоъгълничето и мястото на точката в реда. Всичко това ще поясним със следния пример. Ще се опитаме да „запалим" 135– та точка отляво надясно върху 67-я ред отгоре надолу. Точките от 67 - я ред се намират в деветия ред правоъгълничета. Тъй като всяко правоъгълниче съдържа по седем точки, 135-та точка отляво надясно се намира в двадесетата колона правоъгълничета. След като сме определили мястото на точката, ще изчислим адреса на байта, управляващ реда от седем точки, върху който тя се намира. За целта събираме следните три числа:

адрес на реда на правоъгълничето:	8232	$2028
адрес на колоната на правоъгълничето:	19	$13
относителен адрес на байта спрямо адреса на правоъгълничето:		
2 x 1024 =	2048	$800
	10299	$283B

Относителният адрес на байта спрямо началния адрес на правоъгълничето получаваме, като умножим числото 1024 по номера на реда точки в него. Номера на реда в правоъгълничето намираме, като извадим единица от остатъка, получен от делението на номера на реда точки спрямо началото на екрана на числото 8. За разглеждания случай остатъкът от делението на 67 с 8 е 3. Оттук номерът на реда вътре в правоъгълничето е 3 − 1 = 2.

След като намерим адреса на управляващия байт, необходимо е да определим числото, което трябва да се съдържа в него (фиг.3.2). Първо определяме позицията на точката в 7-битовото поле. За целта разделяме номера на точката спрямо началото на екрана на числото 7. Полученият остатък е равен на номера на позицията, която търсим. За нашия пример той е 2. След това определяме двоичните стойности, които трябва да има всеки бит от байта. Трябва да имаме предвид, че битовете се извеждат в обратен ред и че осмият бит на всеки байт не се извежда.



фиг. 3.1

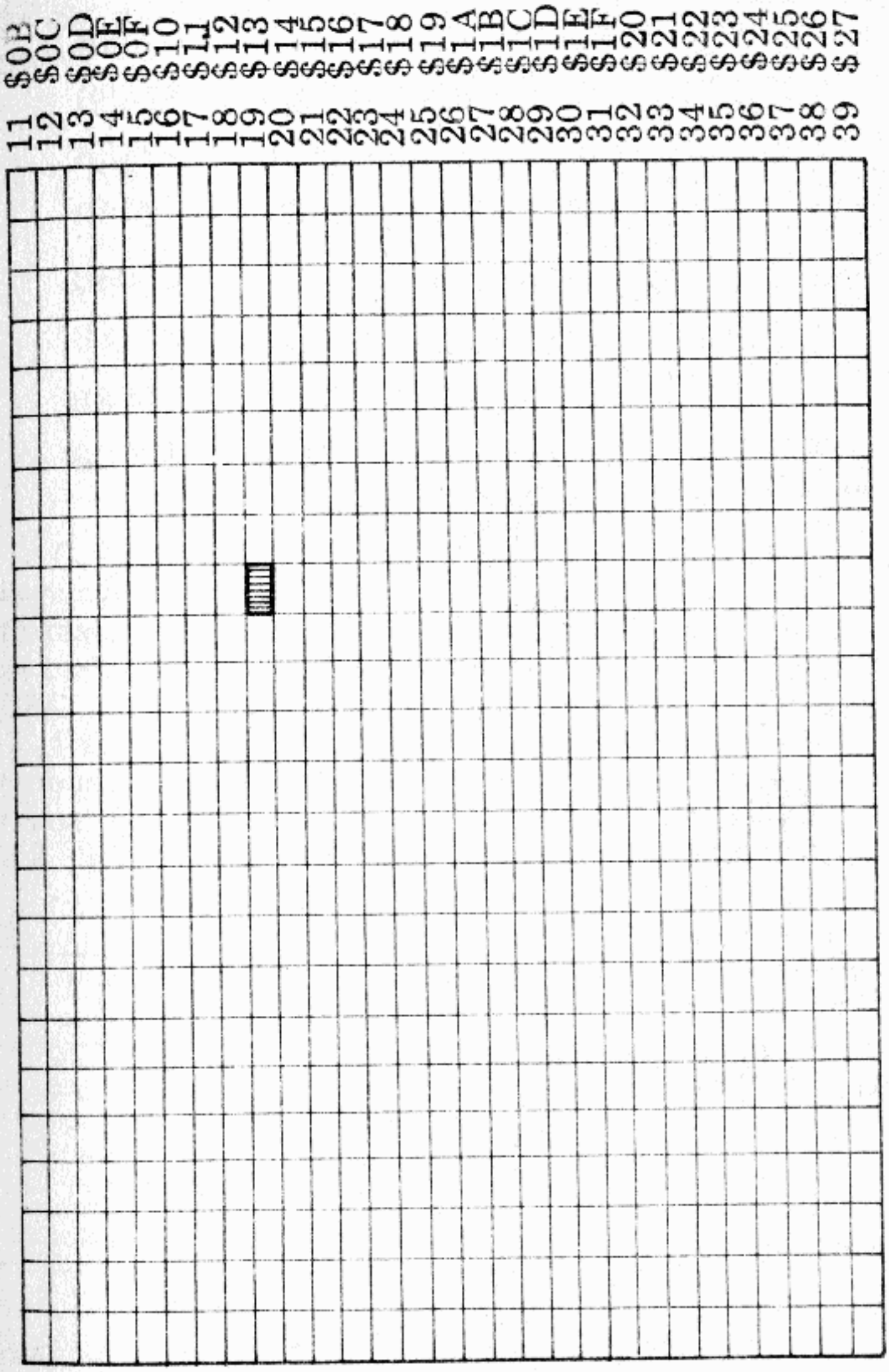
От полученото двоично число изчисляваме шестнадесетичната и десетичната стойност, която трябва да се съдържа в байта. За нашия случай те са съответно 2 и $2.

След тези изчисления сме готови да „запалим" набелязаната точка със следните команди:

```
HGR
POKE 10299,2
```

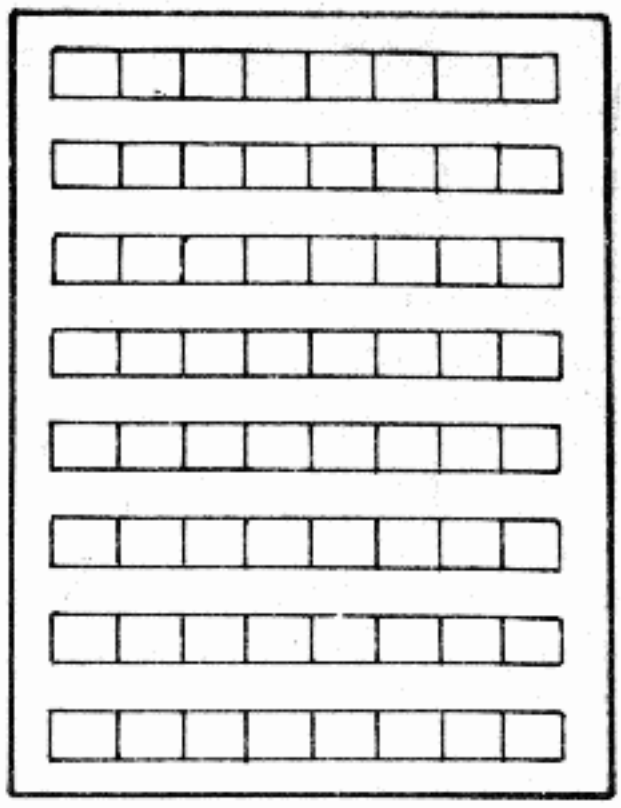
или

```
CALL -151
* 283B:2
```



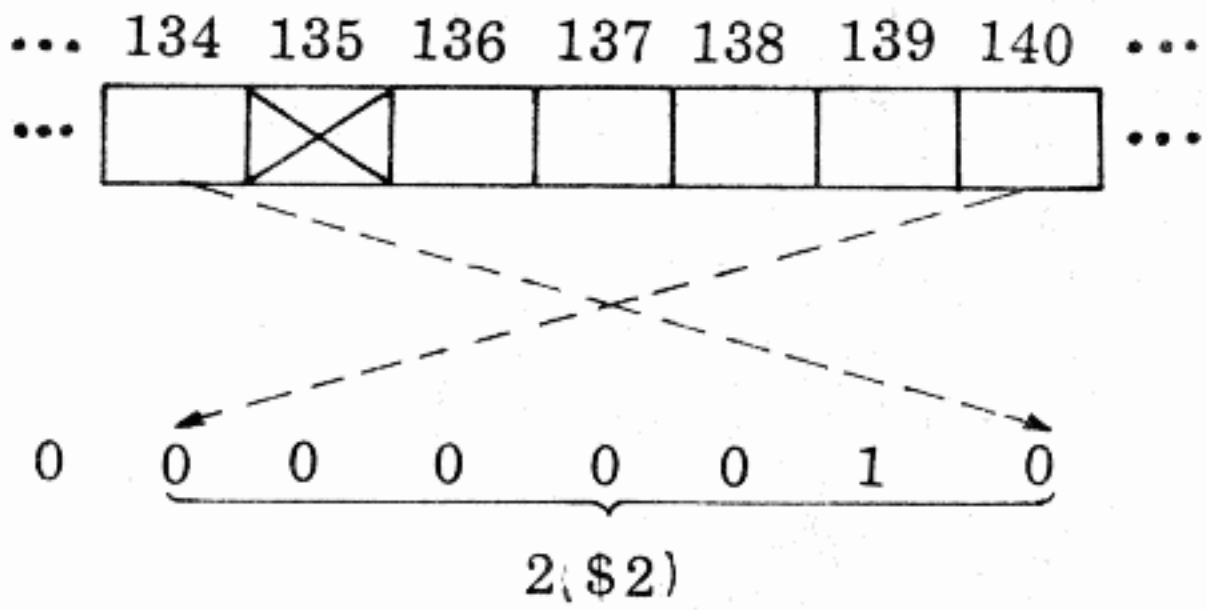
фиг. 3.1 а

Дотук работихме само с една точка от екрана. В следващия пример ще „запалим" едновременно четири точки. Те се намират на ред 115 отгоре надолу и заемат 31,32,33 и 45 позиция отляво надясно. Първите три точки са в едно и също правоъгълниче, а четвъртата е отделно. Затова ще изчислим адресите само на два байта, съдържащи управляващите ги битове. Те са съответно:



8251 = 8251 + 0 $203B
9275 = 8251 + 1024 $243B
10299 = 8251 + 2048 $283B
11323 = 8251 + 3072 $2C3B
12347 = 8251 + 4096 $303B
13371 = 8251 + 5120 $343B
14395 = 8251 + 6144 $383B
15419 = 8251 + 7168 $3C3B

фиг. 3.1 б



фиг. 3.2

адрес на ред:	9000	$2328	9000	$2328
адрес на колоната:	5	$5	7	$7
относителен адрес на байт:	2048	$800	2048	$800
	11053	$2B2D	11055	$2B2F

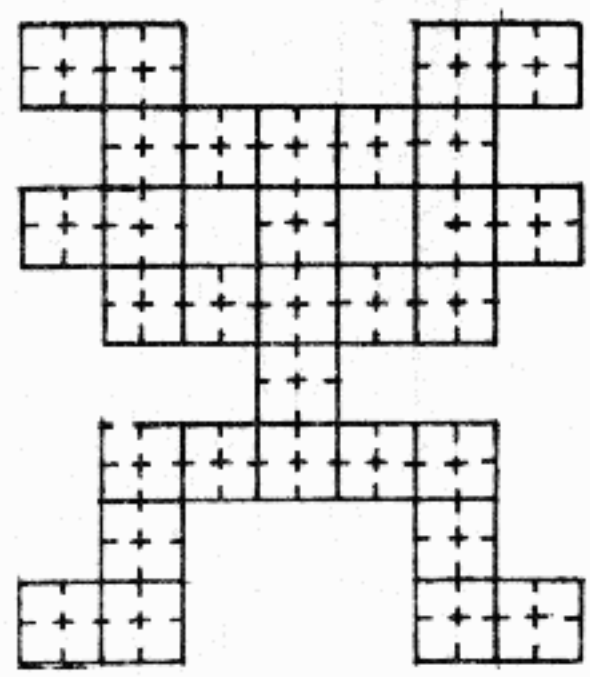
За да определим стойностите, които трябва да запишем в двата байта, ще използуваме следните изображения на полетата, които управляват: − −XXX− − и − −X− − − −. Символа X поставяме на мястото на светеща точка, а „−" − на мястото на несветеща точка. Като отчетем обратния ред на извеждане и добавим „липсващия" осми бит, за съдържанието на съответните байтове получаваме следните стойности: 00011100, 00000100. В десетична и шестнадесетична система те са равни съответно на 28($1C) и 4($4). Получихме необходимите данни,

за да „запалим" четирите бита. Това осъществяваме с командите:

```
HGR
POKE 11053,28
POKE 11055,4
```

или

```
HGR
CALL - 151
* 2 B2D: 1C
* 2B2F: 4
```



фиг. 3.3

След придобития опит дотук сме готови да се опитаме да нарисуваме цяла фигура. Нека това е „извънземното същество", показано на фиг. 3.3. Вижда се, че фигурата на „извънземното" се събира в едно правоъгълниче от картата на паметта в режим BPC.

Първата операция, която трябва да изпълним, е да изчислим стойността на всеки байт, управляващ изображението на фигурата. Отделните стъпки на този процес са показани в табл. 3.2. В резултат са получени десетичните и шестнадесетични

Таблица 3.2

Комбинация на точките	Битова комбинация	Стойност	
		Десетична	Шестнадесетична
× × – – – × ×	0110 0011	99	$63
– × × × × × –	0011 1110	62	$3E
× × – × – × ×	0110 1011	107	$6B
– × × × × × –	0011 1110	62	$3E
– – – × – – –	0000 1000	08	$08
– × × × × × –	0011 1110	62	$3E
– × – – – × –	0010 0010	34	$22
× × – – – × ×	0110 0011	99	$63

стойности на съответните байтове. Изчисляваме относителните адреси на байтовете вътре в правоъгълничето, което ги съдържа, по правилата, описани в първия пример и в съответствие с картата на паметта за режим BPC. Абсолютните адреси на байтовете зависят от координатите на правоъгълничето, в което искаме да поместим фигурата. Програма 3.2 отчита това обстоятелство и чертае „извънземното" във всяко правоъгълниче, което изберем от смесения екран на режим BPC.

```
 1  REM **ПРОГРАМА 3.2**
 2  REM **РИСУВАНЕ НА „ИЗВЪНЗЕМНО" В РЕЖИМ ВРС**
10  REM *НАЧАЛНО УСТАНОВЯВАНЕ*
20  HGR : HOME : DIM AP%(20)
30  FOR I = 1 TO 20: READ AP%(I):NEXT I
40  REM *ИЗБОР НА РЕД И КОЛОНА*
50  VTAB 23
60  INPUT "НОМЕР НА РЕД ?(1 – 20)";HP$
70  HP% = VAL (HP$)
80  IF HP% < 1 OR HP% > 20 THEN PRINT  "":GOTO 60
90  VTAB 24
100  INPUT "НОМЕР НА КОЛОНА? (1 – 40)"; HK$
110  HK% = VAL (HK$)
120  IF HK% < 1 OR HK% > 40 THEN PRINT  "":GOTO 90
130  REM *ИЗВЕЖДАНЕ НА ИЗОБРАЖЕНИЕТО*
140  OA% = AP%(HP%) + HK% – 1
150  POKE OA%,99
160  POKE OA% + 1024,62
170  POKE OA% + 2048,107
180  POKE OA% + 3072,62
190  POKE OA% + 4096,8
200  POKE OA% + 5120,62
210  POKE OA% + 6144,34
220  POKE OA% + 7168,99
230  REM * ПРОВЕРКА ЗА КРАЙ *
240  PRINT "ДРУГА ФИГУРА? (Д/Н)": GET N$
250  IF N$ > < "Н" GOTO 60
260  TEXT: HOME: END
265  REM *НАЧАЛНИ АДРЕСИ*
270  DATA 8192,8320,8448,8576,8704
280  DATA 8832,8960,9088.8232,8360
290  DATA 8488,8616,8744,8872,9000
300  DATA 9128,8272,8400,8528,8655
```

За целта в ред 20 се избира режим ВРС със смесен екран на изображението и маркерът се установява в началото на първия ред от текстовата част на екрана. След това се дефинира целочисленият масив AP% с двадесет елемента и в ред 30 в него се зареждат адресите на редовете от картата на паметта. Те са записани в команди DATA в редове 270 – 300. С редове 60 – 120 се задават номерът на реда HP% и номерът на колоната HK%, определящи правоъгълничето, в което искаме да разположим фигурата. С ред 140 се изчислява неговият основен (базов) адрес OA%. С редове 160 – 220 се изчисляват абсолютните адреси на байтовете, определящи изображението на фигурата, като към основния адрес на правоъгълничето се добавя относителният адрес на всеки байт.

3.3. БИТ ЗА ЦВЕТНОСТ

Хубавият цвят подобрява изображението, разграничава отделните елементи и добавя живот на екрана. Електронните схеми на Правец-82 са конструирани да реализират 6 цвята в режим ВРС. Те са: ЧЕРНО, ЗЕЛЕНО, СИНО, БЯЛО, ВИОЛЕТОВО и ОРАНЖЕВО. Има специални програмни похвати, наречени „възбуждане" или „трептене", чрез които могат да се постигнат до 20 различни оттенъци на цветовете. В тази точка ще разгледаме конструктивните възможности и програмните похвати за реализиране на цветовете в режим ВРС на Правец-82.

Осмият бит на байтовете от графичните страници на режим ВРС не се извежда на екрана, а съдържа информация за цвета на останалите битове. Поради това той се нарича бит за цветност. Неговата стойност определя коя група цветове може да съдържа останалата част от байта. Ако е нула, останалите седем бита могат да съдържат цветовете от първа група – ЧЕРНО I, ЗЕЛЕНО, ВИОЛЕТОВО и БЯЛО I. Ако е единица, съответният байт може да съдържа цветовете от втора група – ЧЕРНО II, ОРАНЖЕВО, СИНО и БЯЛО II. Тези цветове могат да варират в зависимост от различията на използуваните видеомонитори, но винаги могат да се постигнат чрез подходяща настройка.

Основните правила, които трябва да се имат предвид при управление на цветовете на отделните точки на екрана в режим ВРС, са следните:

1. Ако точката се намира в четна колона, може да има виолетов или син цвят.

2. Ако точката се намира в нечетна колона, може да има зелен или оранжев цвят.

3. Колоните от точки на екрана се броят отляво надясно от 0 до 279.

4. Всеки от двата възможни цвята за една точка от екрана се избира чрез осмия бит (бита за цветност) на управляващия байт. Точната зависимост между състоянието на бита за цветност и избрания цвят се определя по табл. 3.3.

Таблица 3.3

Състояние на бита за цветност	Четна колона	Нечетна колона
0	виолетово	зелено
1	синьо	оранжево

5. Точката има черен цвят, когато управляващият я бит е установен в нула.

6. Бял цвят се постига, като се установят едновременно в единица два съседни бита.

Във версията на БЕЙСИК за Правец-82 цветовете за режим ВРС се задават чрез HCOLOR и числата от 0 до 7 по следния начин:

0 – черно I	4 – черно II
1 – зелено	5 – оранжево
2 – виолетово	6 – синьо
3 – бяло I	7 – бяло II

Трябва да се има предвид, че в някои случаи могат да се получат различия за цветовете зелено, оранжево, виолетово и синьо в зависимост от използувания видеомонитор.

Ще разгледаме действието на няколко прости програми, като вземем предвид изброените правила.

```
 1  REM  **ПРОГРАМА 3.3**
 2  REM  **ПРИМЕР 1 – УПРАВЛЕНИЕ НА ЦВЕТОВЕТЕ В РЕ-
     ЖИМ  ВРС**
10   HGR
20  HCOLOR = 3:REM БЯЛО I
30  HPLOT 10,0 TO 10,191
35   REM  *ПАУЗА*
40   FOR I = 1 TO 500: NEXT
50   HCOLOR = 4: REM ЧЕРНО II
60   FOR I = 0 TO 191
70   HPLOT 9,I
80   NEXT I
90   END
```

В ред 20 на програма 3.3 е кодиран цвят бяло I. Въпреки това при изпълнението ѝ в лявата част на екрана (колона 10) вместо бяла се изчертава виолетова линия. След пауза от няколко секунди (осъществена от ред 40) цветът на линията се променя от виолетов на син. Тези резултати се обясняват по следния начин. Първата част на програмата (до ред 40) не изчертава бяла линия, тъй като съгласно правило 6 бял цвят се получава само ако два съседни бита са установени в единица. За разглеждания случай в единица се установява само по един бит от всеки ред. От друга страна, този единствен бит винаги носи някакъв цвят в зависимост от колоната, в която се намира и състоянието на бита за цветност. В нашия пример той е виолетов, защото точката се намира в четна колона и битът ѝ за цветност е 0. Той е установен в това положение при изпълнение на ред 20,

където е избрана първата група цветове, към която принадлежи и бяло I.

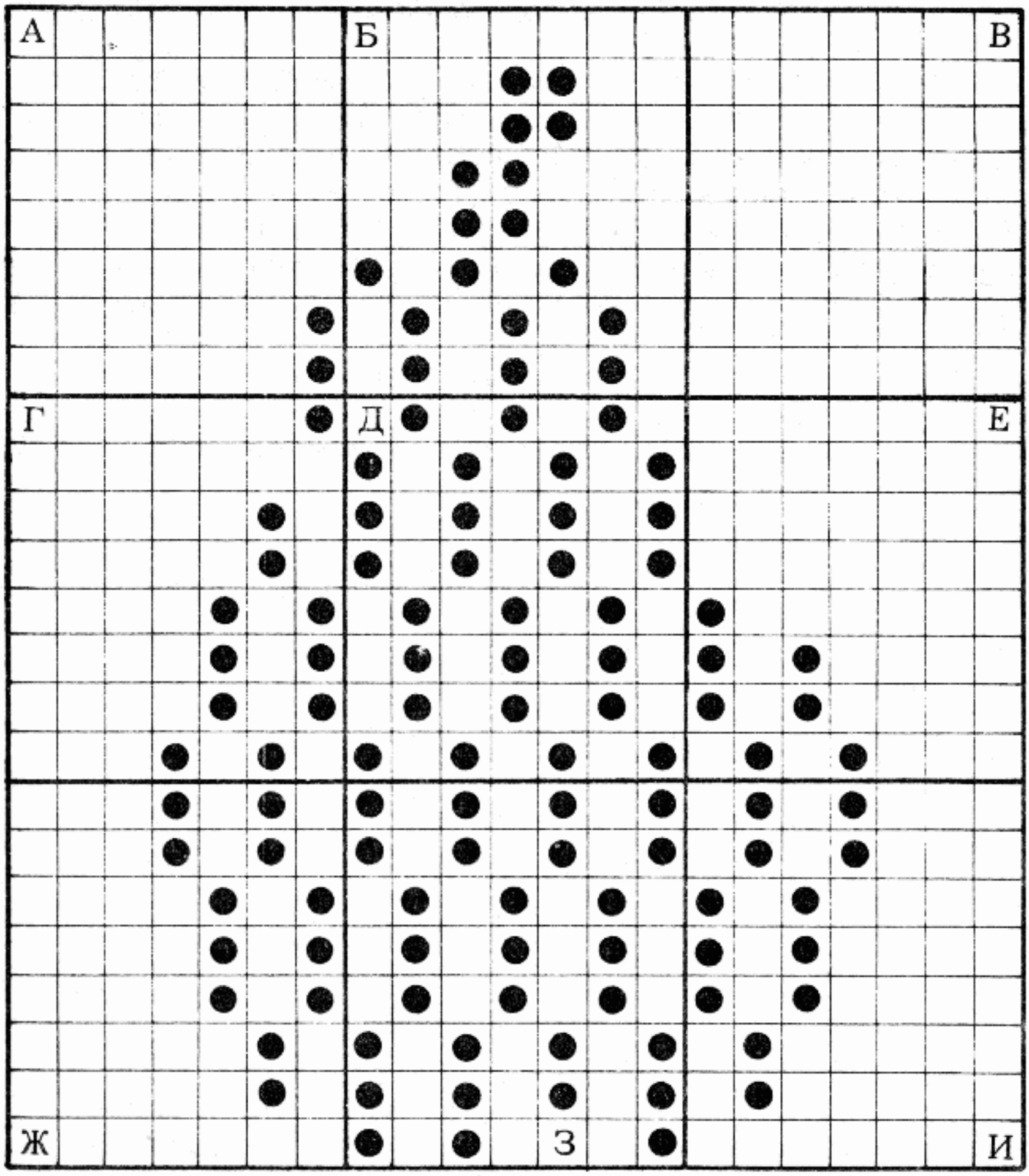
Във втората част на програмата установяваме нов цвят — черно II, и с него изчертаваме линия в девета колона. Тя не се вижда и това е нормално. Тъй като девета и десета колона се управляват от едни и същи байтове, при избора на цвят черно II се променя състоянието на бита за цветност на тази група байтове от 0 в 1. С това се избира втората група цветове, вследствие на което установените в 1 битове на десета колона вече носят син цвят. Ако в ред 50 от програмата вместо черно II установим цвят черно I, няма да променим състоянието на битовете за цветност на байтовете, управляващи десета колона. Тогава нейният цвят няма да се промени и ще остане виолетов.

```
 1  REM **ПРОГРАМА 3.4**
 2  REM **ПРИМЕР 2 – УПРАВЛЕНИЕ НА ЦВЕТОВЕТЕ В РЕ-
    ЖИМ  BPC**
10   HGR
20  HCOLOR = 6: REM СИНЬО
30  FOR I = 1 TO 278
40   HPLOT I,100
50   NEXT I
60   REM *ИЗТРИВАНЕ*
70  HCOLOR = 0: REM ЧЕРНО I
80  FOR I = 2 TO 278 STEP 2
90   HPLOT I,100
100   NEXT I
110   END
```

Програмата 3.4 демонстрира други особености на работата с цветовете в режим BPC. Преди да я изпълним, оставаме с впечатление, че с първата ѝ част (редове 20 – 50) трябва да се изчертае непрекъсната синя линия в ред 100 от екрана, а с втората ѝ част (редове 70 – 100) сините точки от реда да се изтриват през една. Когато я изпълним, виждаме, че синята линия се изчертава нормално, а след това се изтрива напълно. Това е така, защото съгласно правило *1* синя точка може да има само в четните колони. При изчертаването на хоризонталната синя линия в нечетните точки в действителност не се чертае нищо. Поради това, когато във втората част на програмата чертаем с черен цвят в четните колони (ред 90), изтриваме цялата линия.

3.4. ЦВЕТНА ГРАФИКА В РЕЖИМ BPC

Описаните правила за кодиране на цветовете в режим BPC ще използуваме, за да нарисуваме крушата, показана на фиг. 3.4. Тя



фиг. 3.4

е разделена на 9 части, означени с буквите от А до И. В табл. 3.4 е показано преобразуването на точковите модели (комбина-

Таблица 3.4

Сектор	Точков модел	Битов модел	Стойност	
			Шестн.	Десет.
А	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
	– – – – – – – ×	01000000	40	64
	– – – – – – – ×	01000000	40	64

Таблица 3.4_продължение 1

Сектор	Точков модел	Битов модел	Стойност	
			Шестн.	Десет.
Б	– – – – – – – –	00000000	00	0
	– – – × × – –	00011000	18	24
	– – – × × – –	00011000	18	24
	– – × × – – –	00001100	0C	12
	– – × × – – –	00001100	0C	12
	× – × – × – –	00010101	15	21
	– × – × – × –	00101010	2A	42
	– × – × – × –	00101010	2A	42
В	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
Г	– – – – – – – ×	01000000	40	64
	– – – – – – – –	00000000	00	0
	– – – – – × –	00100000	20	32
	– – – – – × –	00100000	20	32
	– – – – × – ×	01010000	50	80
	– – – – × – ×	01010000	50	80
	– – – – × – ×	01010000	50	80
	– – – × – × –	00101000	28	40
Д	– × – × – × –	00101010	2A	42
	× – × – × – ×	01010101	55	85
	× – × – × – ×	01010101	55	85
	× – × – × – ×	01010101	55	85
	– × – × – × –	00101010	2A	42
	– × – × – × –	00101010	2A	42
	– × – × – × –	00101010	2A	42
	× – × – × – ×	01010101	55	85
Е	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
	– – – – – – – –	00000000	00	0
	× – – – – – –	00000001	01	1
	× – × – – – –	00000101	05	5
	× – × – – – –	00000101	05	5
	– × – × – – –	00001010	0A	10
Ж	– – – × – × –	00101000	28	40
	– – – × – × –	00101000	28	40
	– – – – × – ×	01010000	50	80
	– – – – × – ×	01010000	50	80
	– – – – × – ×	01010000	50	80
	– – – – – × –	00100000	20	32
	– – – – – × –	00100000	20	32
	– – – – – – –	00000000	00	0

Таблица 3.4 —продължение 2

Сектор	Точков модел	Битов модел	Стойност	
			Шестн.	Десет.
З	× — × — × — ×	01010101	55	85
	× — × — × — ×	01010101	55	85
	— × — × — × —	00101010	2A	42
	— × — × — × —	00101010	2A	42
	— × — × — × —	00101010	2A	42
	× — × — × — ×	01010101	55	85
	× — × — × — ×	01010101	55	85
	× — × — × — ×	01010101	55	85
И	— × — × — — —	00001010	0A	10
	— × — × — — —	00001010	0A	10
	× — × — — — —	00000101	05	5
	× — × — — — —	00000101	05	5
	× — × — — — —	00000101	05	5
	— × — — — — —	00000010	02	2
	— × — — — — —	00000010	02	2
	— — — — — — —	00000000	00	0

ции) на отделните части (сектори) на изображението в десетични стойности. Освен опашката, която е бяла, останалата част на крушата е изобразена с точки, разположени през една колона. Това подреждане се налага от изискването всеки цвят в режим BPC да се разполага само в четни или само в нечетни колони. След като цветното изображение на фигурата е кодирано с числа, можем да го нарисуваме на екрана. Това извършваме с програма 3.5.

```
 1  REM **ПРОГРАМА 3.5**
 2  REM **ЦВЕТНА РИСУНКА ВЪВ BPC**
 5  REM *НАЧАЛНО УСТАНОВЯВАНЕ*
10  OTM% = 1024: DIM AP%(20)
20  FOR I = 1 TO 20: READ AP%(I):NEXT
25   HOME
30   HGR: VTAB (22)
40   REM *ВЪВЕЖДАНЕ НОМЕР НА РЕД*
50   INPUT „НОМЕР НА РЕД? (1 — 18)";HP$
60   HP% = VAL (HP$): IF HP% < 1 OR HP% > 18
     THEN PRINT CHR$(7): GOTO 50
70   REM *НОМЕР НА КОЛОНА*
80   INPUT "НОМЕР НА КОЛОНА? (1 — 38)"; HK$
90  HK% = VAL (HK$): IF HK% < 1 OR HK% > 38 THEN PRINT
    CHR$ (7): GOTO 80
100   REM * ИЗВЕЖДАНЕ НА ИЗОБРАЖЕНИЕТО*
110   FOR I = 0 TO 2
120   FOR J = 0 TO 2
130   OA% = AP% (HP% + I) + (HK% — 1) + J
```

```
140  FOR K = 1 TO 8
150  READ CT% : POKE OA% + (K - 1) * OTM%,CT%
160   NEXT K,J,I
170   END
180   REM *НАЧАЛНИ АДРЕСИ*
190   DATA 8192,8320,8448,8576,8704
200   DATA 8832,8960,9088,8232,8360
210   DATA 8488,8616,8744,8872,9000
220   DATA 9128,8272,8400,8528,8656
230   REM *ДАННИ ЗА ИЗОБРАЖЕНИЕТО*
240   DATA 0,0,0,0,0,0,64,64
250   DATA 0,24,24,12,12,21,42,42
260   DATA 0,0,0,0,0,0,0,0
270   DATA 64,0,32,32,80,80,80,40
280   DATA 42,85,85,85,42,42,42,85
290   DATA 0,0,0,0,1,5,5,10
300   DATA 40,40,80,80,80,32,32,0
310  DATA 85,85,42,42,42,85,85,85
320   DATA 10,10,5,5,5,  2,2,0
```

Тя дава възможност за избор на мястото, върху което се помества фигурата. За целта е необходимо да се посочат координатите на горния ляв ъгъл от областта, в която се разполага фигурата. С ред 10 от програмата в променливата OTM% се зарежда стойността 1024. Това е отместването между адресите на отделните редове точки в едно правоъгълниче. Едномерният масив AP%, дефиниран с размер 20 елемента, съдържа началните адреси на всеки ред правоъгълничета. Те са дадени в лявата част на картата на паметта от фиг. 3.1. В ред 20 масивът се зарежда с тези стойности. Те се четат от блокове данни, разположени в редове 190 – 220. С ред 30 се установява режим BPC със смесен екран на изображението и се позиционира маркерът в текстовата му област. Следващата част от програмата (редове 50 – 90) осъществява диалог за определяне координатите на правоъгълничето, което съдържа левия горен край на фигурата. При проверката за вярно въведен номер на ред се отчита, че цялата фигура е разположена върху три правоъгълничета във вертикална посока. Поради това се проверява за най-голям възможен номер 18, а не 20. По аналогични съображения се извършва проверка за най-голям номер на колона 38, а не 40.

С редове 110 – 160 се извършва действителният запис на информацията за изображението в избраната област от паметта. Цифровите данни от табл. 3.4 са записани в редове 240 – 320 на програмата. Те се четат последователно и чрез подходящо индексиране с променливите K, I и J се поместват в съответните байтове на паметта. Променливата I индексира

номерата на редовете правоъгълничета, J – номерата на колоните, а K – номерата на редовете точки вътре във всяко правоъгълниче. Тъй като нашата фигура се помества върху област от екрана с размер 3 на 3 правоъгълничета, индексите I и J се променят в границите от 0 до 2, а K се променя от 1 до 8.

Ако изпълним програмата няколко пъти и зададем различни места, върху които се разполага фигурата, ще видим, че тя променя цветовете си в зависимост от избрания номер колона. Именно затова, когато кодирахме различните части на кръшата, ние не ги свързахме с определен цвят.

Друга особеност на програмата е необходимостта винаги да разполагаме фигурата през цяло число правоъгълничета спрямо началото на екрана. Например може да искаме да започнем да чертаем на три и половина правоъгълничета спрямо левия край на екрана, а сме задължени да зададем четири. Този недостатък се избягва, като използуваме програмни средства, които ще разгледаме в следващите глави.

3.5. ПОВЕЧЕ ОТ ШЕСТ ЦВЯТА

Съгласно с описанието на версията на БЕЙСИК за „Правец-82" в графичен режим ВРС разполагаме с шест цвята. Те бяха използувани в разгледаните дотук примери. Съществуват програмни похвати, които при повече въображение позволяват да постигнем изкуствено допълнителни цветове. Един от тях се нарича „трептене". За да го обясним, нека изпълним програмата 3.6. В горния ляв ъгъл на екрана се появява правоъгълник, който променя постоянно цвета си. При това освен през основните шест цвята той преминава и през допълнителни оттенъци. Скоростта на промените може да се регулира, като се изменя крайната стойност на цикъла в ред 100. Размерът на правоъгълника се определя от променливата РМ.

```
1   REM  * * ПРОГРАМА 3.6 * *
2   REM  * * СМЕСВАНЕ НА ЦВЕТОВЕ * *
10  HGR :PM = 50
20  FOR I = 1 TO 7
30  FOR J = 1 TO 7
40  FOR Y = 0 TO PM STEP 2
50  HCOLOR = I
60  HPLOT 0,Y TO PM,Y
70  HCOLOR = J
80  HPLOT 0,Y + 1 TO PM,Y + 1
90  NEXT Y
100  FOR K = 1 TO 200: NEXT K
110  NEXT J
120  NEXT I
130  END
```

Действието на програмата е следното. Изчертават се последователно следващи двойки хоризонтални линии с различни цветове, които се задават от променливите I и J. След като се запълни правоъгълникът по този начин, единият цвят се сменя. Така цикълът продължава. В резултат виждаме основните цветове – зелено, синьо, виолетово, оранжево, черно и бяло. Освен това към тях се добавят нови цветове, например светлозелено и ярковиолетово. Те се получават от смесването на два основни цвята. Съществуват множество комбинации, чрез които се получават нови цветове. Например можем да смесваме черно с останалите цветове, за да получим тъмносиньо или тъмнооранжево. Можем да смесваме цвят, получен от смесването на два цвята, с основен цвят и т.н.

Описаният метод има известни ограничения, които трябва да се имат предвид. Те се проявяват, когато смесваме цветове в хоризонтална посока по продължение на реда. Такъв е случаят с програмата 3.7, където се смесват оранжево и синьо.

```
1   REM * * ПРОГРАМА 3.7 * *
2   REM * * СМЕСВАНЕ НА ЦВЕТОВЕ ПО ДЪЛЖИНА НА РЕДА * *
10  HOME : HGR
20  FOR X = 0 TO 278 STEP 2
30  HCOLOR = 6: HPLOT X,Y
40  FOR I = 1 TO 200: NEXT I
50  HCOLOR = 5: HPLOT X + 1,Y
60  FOR I = 1 TO 200: NEXT I
70  NEXT X
80  END
```

В началото на изпълнението ѝ първата точка се начертава със син цвят. При появата на втората, която трябва да е оранжева, двете точки стават бели. Останалите точки също се извеждат бели независимо от цвета, който сме задали в програмата. Това се получава в съответствие с правило *6* за кодиране на цветовете в режим BPC. Според него, ако два съседни бита в един байт са установени в единица, те извеждат бял цвят. Поради това, ако искаме да извеждаме редуващи се цветове по продължение на реда трябва да оставяме най-малко по една черна точка между тях.

Освен това обърнете внимание, че програмата извежда сините точки в четните, а оранжевите – в нечетните позиции. Ако сменим кодовете на цветовете в редове 30 и 50, на екрана няма да се появи нищо. Това е резултат от правила *1* и *2* за управление на цветовете.

С това не се изчерпват ограниченията, които трябва да спазваме.

```
1   REM * * ПРОГРАМА 3.8 * *
2   REM * * СМЕСВАНЕ НА ЦВЕТОВЕ ОТ РАЗЛИЧНИ ГРУПИ * *
10   HGR
20   FOR X = 0 TO 276 STEP 4
30   HCOLOR = 2: HPLOT X,Y
40   FOR I = 1 TO 200: NEXT I
50   HCOLOR =6; HPLOT X + 2,Y
60   FOR I = 1 TO 200: NEXT I
70   NEXT X
80   END
```

Програмата 3.8 демонстрира още едно явление, което трябва да се има предвид. Тя е създадена да чертае виолетова точка в колона 0, синя точка в колона 2, виолетова в 4, синя в 6 и т.н. до края на реда. Когато я изпълним, виолетовата точка (цвят от първа група) се появява в колона 0. След това в колона 2 се появява синя точка. Тъй като синият цвят е от втора група, битът за цветност на съответния байт се установява в единица, с което се избира втората група цветове. В резултат на това и точката от колона 0 става синя. При извеждане на точка в колона 4 отново се избира първа група цветове. С това предните точки стават виолетови. Извеждането на точка в колона 6 избира втора група цветове и отново всички точки стават сини. Колона 8 се управлява от следващия байт, така че появата на виолетова точка в нея не променя предните точки и те остават сини. В колона 10 се извежда синя точка, която установява син цвят и в колона 8. Появата на точка в колона 12 установява всички точки от втория байт във виолетов цвят. Колона 14 се управлява от нов байт, поради което точката, появяваща се в нея, не променя предните цветове. В резултат от изпълнението на програмата се начертава хоризонтална линия, в която се редуват точки с виолетов и син цвят.

Явлението, което описахме дотук, е характерно за графичния режим ВРС и се нарича несъвместимост. То се получава при неподходящо използуване на цветовете от двете групи в рамките на един и същ байт. Ние не можем да смесваме зелено и виолетово съответно с оранжево и синьо в един байт. Това трябва да се има предвид при планиране на цветовете на фигурите от изображението в режим ВРС.

3.6. ГРАФИЧЕН РЕДАКТОР ЗА РЕЖИМ ВРС

За чертане на фигури в режим ВРС също могат да се използуват графични редактори. Те заместват писането на програми с многократно използуване на командата HPLOT. Програма на графичен редактор за режим ВРС се съдържа в приложената ди-

скета. Тя е записана с името ПР 3.9 и е подобна на програмата на редактора за режим НРС. Тук също се използуват клавишите И/I, Й/J, К/К и М/М за управление на мигащ маркер, който при движението си по екрана оставя следа с предварително избран цвят. Чрез клавиша ОСВ се влиза в меню, което съдържа основните функции на редактора:

1 – редактиране;
2 – запис;
3 – четене (зареждане);
4 – изчистване на екрана;
5 – край.

В режима на редактиране екранът се разделя на две части. Горната се използува за чертане, а долната (последните четири реда) – за извеждане на съобщения. Най-вляво се появява кодът на цвета, с който се чертае в момента. Той може да се променя от 0 до 7, като се натиска последователно клавишът Ц/С. В средата на зоната за съобщения се явяват надписите НЕ ЧЕРТАЕ или ЧЕРТАЕ. Режимът, в който не се чертае, се различава от съответния режим на редактора за НРС. Тук маркерът изобщо не се извежда на екрана. Причината за това е, че макар и да не чертаем в определена точка от екрана, появата на мигащия маркер може да промени състоянието на бита ѝ за цветност. С това има вероятност да се променят цветовете на съседните точки, управлявани от същия бит. Тази особеност трябва да се има предвид и в режим на чертане. Режимите „чертае" и „не чертае" се избират чрез еднократно натискане на клавиша Д/D.

Третото съобщение показва интервала, през който се изписват отделните точки. Съществуват три възможности. Точките могат да се чертаят една до друга, през пет и през десет точки. Отделните възможности се избират, като се натиска последователно клавишът Р/R. Съответното число се извежда срещу надписа РАЗСТОЯНИЕ.

В режим 2 – запис цялото съдържание на графичната страница за режим ВРС, се записва в двоичен файл върху дискетата.

В режим 3 – четене съдържанието на двоичен файл, се зарежда в графичната страница и съответното изображение се появява на екрана.

И в двата случая на редактора трябва да зададем името на файла, с който искаме да обменим информация.

В режим 4 – изчистване на екрана текущото изображение се изтрива от екрана.

ГЛАВА 4. ФОРМИ И ТАБЛИЦИ НА ФОРМИ

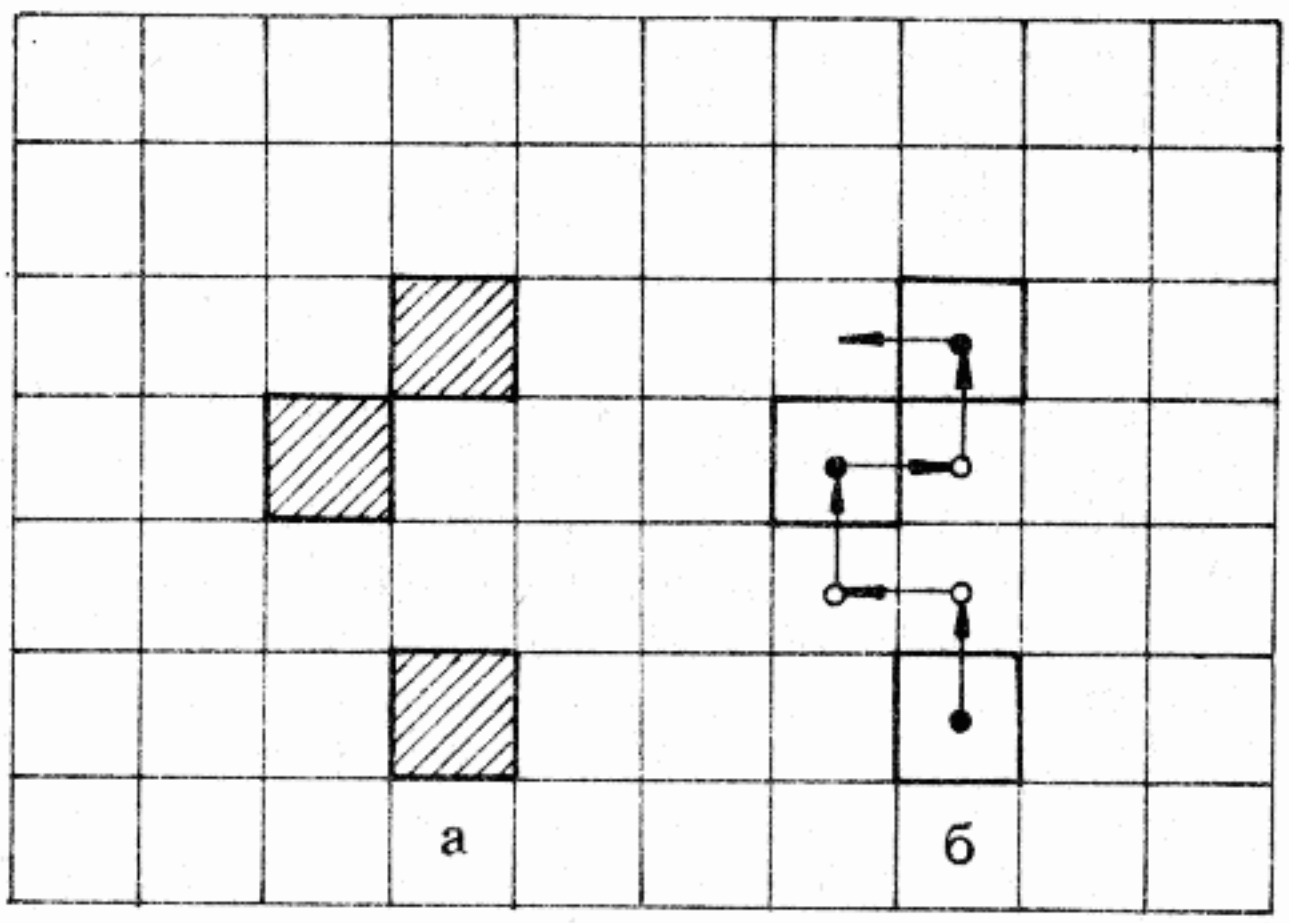
В тази глава ще разгледаме възможностите на микрокомпютъра Правец-82 за работа с графични форми и таблици на форми. Понятието форма тук има специфичен смисъл. Докато за програмиста то означава някаква фигура, например ракета, кола, пеперуда и др., за микрокомпютъра *формата* е специално подредено множество от числа, които определят изображението на фигурата върху екрана.

Въпреки че използването на форми отначало изглежда отегчително и трудопоглъщащо, на практика то облекчава значително чертането на графични фигури и е особено полезно при реализиране на движещи се изображения. Графичните форми могат да се използват успешно и за изписване на текст в графичен режим с висока разделителна способност. След като дадена фигура се кодира веднъж като поредица от числа в съответствие с правилата за създаване на таблица на формите, тя лесно може да се извежда върху екрана. При това не е нужно да се изчисляват поотделно адресите, в които се разполагат данните, определящи всяка точка от изображението на фигурата.

4.1. КОДИРАНЕ НА ФОРМИ

За да начертаем определена форма, използуваме последователност от вектори, които показват на компютъра дали да се изчертае текущата точка от екрана и в коя посока да се премести, за да се премине към следващата точка. Всеки вектор се изобразява с малка стрелка и кръгче в началото ѝ

Ще започнем със съвсем прост пример. На фиг. 4.1*а* е начертана форма, която заема само три точки от екрана, а на фиг. 4.1*б* са дадени векторите, които я описват. Запълненото кръгче в началото на вектора показва, че съответната точка трябва да се изчертае. Празното кръгче показва, че точката трябва да се прескочи. Стрелката сочи посоката към следващата точка. Важно е да запомните, че изчертаването или прескачането на точка се извършва преди преминаването към следващата позиция. Ще използуваме буквата Ч, за да означим точка, която трябва да се изчертае, и П за точка, която трябва да се прескочи. Освен това с Г ще означим, че се движим нагоре, с Д — надясно, с Л — наляво и с Н — надолу. При тази постановка да обясним фиг. 4.1*б*.



фиг. 4.1

Първият вектор показва, че точката, която представя, трябва да се изчертае и да се премине нагоре. Това ще означим с ЧГ. Вторият прескача съответната точка и се премества наляво (ПЛ). Третият прескача точката и се премества нагоре (ПГ), четвъртият, петият и шестият съответно: чертае и преминава надясно (ЧД), прескача и преминава нагоре (ПГ) и чертае и преминава наляво (ЧЛ). В резултат е получен списъкът от вектори: ЧГ, ПЛ, ПГ, ЧД, ПГ и ЧЛ. Поради това, че векторът чертае винаги, преди да се премести, посоката на последния вектор е избрана произволно. Важната характеристика за него е командата за чертане, с която се запълва последната точка.

Сега микрокомпютърът може да начертае формата върху екрана, като изпълни следната последователност от инструкции: чертай – нагоре, прескочи – наляво, прескочи – нагоре, чертай – надясно, прескочи – нагоре и чертай – наляво. Необходимо е да кодираме всяка от тези инструкции чрез двоични числа по следния начин:

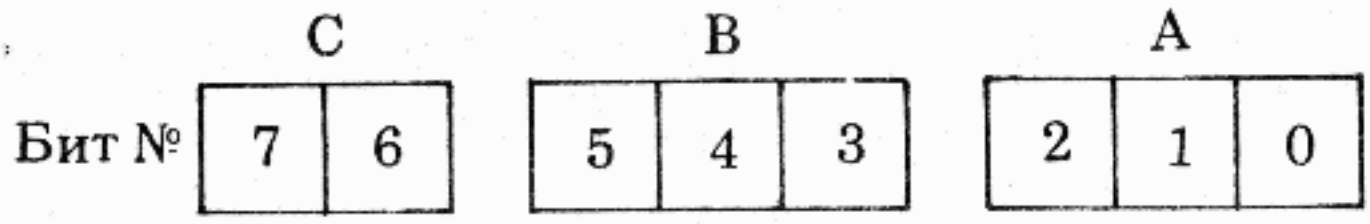
чертай: 1
прескочи: 0

нагоре: 00
надясно: 01
надолу: 10
наляво: 11

За да кодираме първия вектор (ЧГ), използуваме 1 за „чертане" и 00 за „нагоре". Така получаваме 100. Вторият вектор (ПЛ) дава 011, а третият (ПГ) – 000. Накрая за шестте вектора и съответните им кодове получаваме:

ЧГ: 100 ПЛ: 011 ПГ: 000 ЧД: 101 ПГ: 000 ЧЛ: 111

За да съобщим тези стойности на компютъра, трябва да ги поместим в байтове. За тази цел всеки байт се разделя на три части (секции) A, B и C (фиг. 4.2).



фиг. 4.2

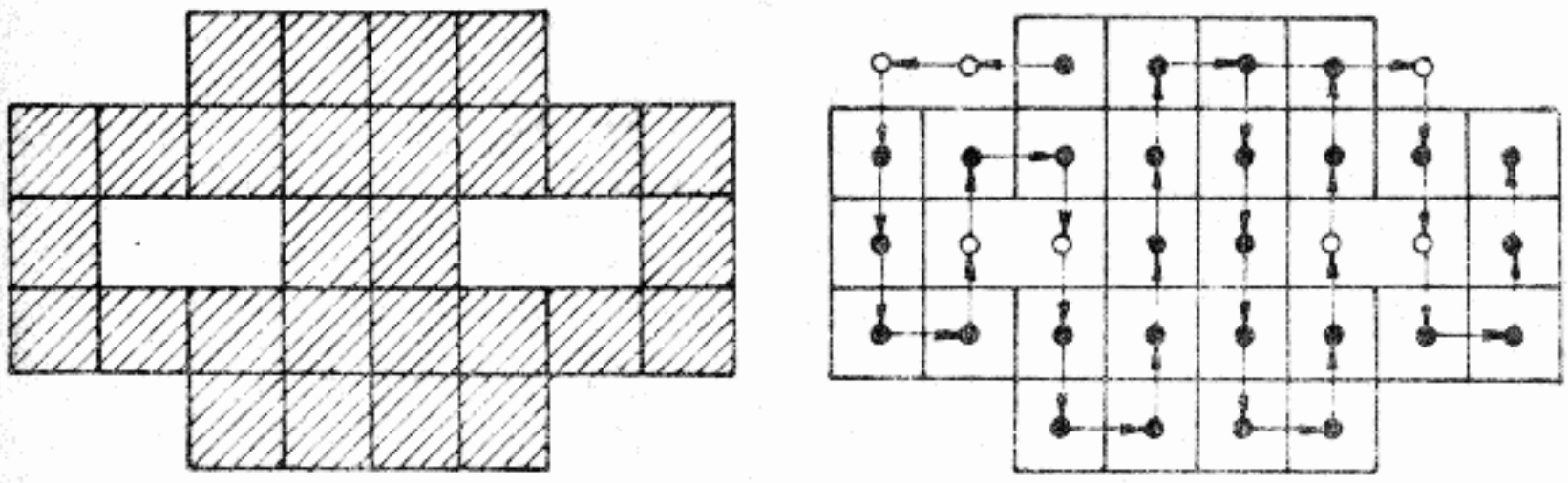
Ще започнем, като запишем първия код на вектор в част А, втория в В и третия (ако е възможно) в С. Не е позволено да разделяме кодовете на части и да записваме остатъка им в следващ байт. Част С има само два бита. Затова тя не може да съдържа цял код. Логично е да приемем, че част С не може да се използува и да решим да я оставяме винаги празна (00). Има случаи обаче, в които тя може да се използува. Ако най-левият бит на кода, който трябва да се запише в част С, е 0, той може да се изпусне и да се използуват само десните два бита. Например 10 000 101 означава да се изпълни ЧЛ (101), ПГ (000) и ПН (10).

Всеки вектор, който се записва в част С, трябва да е вектор за прескачане. Когато се изчертава формата, двата нулеви бита в левия край на байта се пренебрегват. Всъщност всяка нулева секция на даден байт се пренебрегва, ако всички следващи части на този байт са нулеви. Например при 00 000 011 текущата точка ще се прескочи и ще се премине с една точка наляво, след което остатъкът от байта ще се пренебрегне и ще се започне изпълнението на кодовете от следващия байт. От това следва, че в секция С никога не може да се изпълни векторът „нагоре" (00), тъй като той винаги ще се пренебрегва. Например за байта: 00 111 000 ще се изпълнят инструкциите прескочи – нагоре (000), чертай – наляво (111) и ще се пренебрегне част С. В този случай векторът „прескочи – нагоре" трябва да се запише в част А на следващия байт.

На фиг. 4.3 са показани групираните вектори на формата, която създадохме. Частите С на първите три байта са установени в нула, поради което се пренебрегват. Четвъртият байт е запълнен с нули, които маркират края на формата. Чертането на всяка форма се преустановява, когато се срещне нулев байт. Поради това, ако ви се наложи да сложите код 000 в части А и В, вие задължително трябва да запишете ненулев код в част С.

Байт №	C	B	A	C	B	A	Шестн.код
1		ПЛ	ЧГ	00	011	100	1C
2		ЧД	ПГ	00	101	000	28
3		ЧЛ	ПГ	00	111	000	38
4				00	000	000	00

фиг. 4.3



а б

фиг. 4.4

Макар и да не вярвате в неидентифицирани летящи обекти, да се опитаме да създадем формата на един от тях. На фиг. 4.4*а* е нарисувана „двойната чиния", а на фиг. 4.4*б* е показано векторното ѝ описание.

За да я изобразим върху екрана, необходимо е да начертаем всяка от съставящите я точки. Последователността, в която ще извършим това, не е съществена, но промяната на посоката на векторите води до промяна на числовите стойности на формата. Затова, ако следваме означената на фиг. 4.4*б* последователност на векторите и напишем списъка на инструкциите, които ги описват, ще получим:

ЧЛ, ПЛ, ПН, ЧН, ЧН, ЧД, ЧГ, ПГ, ЧД, ЧН,
ПН, ЧН, ЧД, ЧГ, ЧГ, ЧГ, ЧГ, ЧД, ЧН, ЧН,
ЧН, ЧН, ЧД, ЧГ, ЧГ, ПГ, ЧГ, ЧД, ПН, ЧН,
ПН, ЧД, ЧГ, ЧГ, ЧГ

Следващата стъпка е да кодираме инструкциите в байтове по следния начин:

С	В	А	С	В	А	шестн.	десет.
ПН	ПЛ	ЧЛ	10	011	111	9F	159
– –	ЧН	ЧН	00	110	110	36	54
– –	ЧГ	ЧД	00	100	101	25	37
– –	ЧД	ПГ	00	101	000	28	40
– –	ПН	ЧН	00	010	110	16	22
– –	ЧД	ЧН	00	101	110	2E	46
– –	ЧГ	ЧГ	00	100	100	24	36
– –	ЧГ	ЧГ	00	100	100	24	36
– –	ЧН	ЧД	00	110	101	35	53
– –	ЧН	ЧН	00	110	110	36	54
– –	ЧД	ЧН	00	101	110	2E	46
– –	ЧГ	ЧГ	00	100	100	24	36

— —	ЧГ	ПГ	00	100	000	20	32
— —	ПН	ЧД	00	010	101	15	21
— —	ПН	ЧН	00	010	110	16	22
— —	ЧГ	ЧД	00	100	101	25	37
— —	ЧГ	ЧГ	00	100	100	24	36

Получените байтове преобразуваме в шестнадесетични и десетични числа. По този начин получаваме цифровото представяне на „двойната чиния". Сега е необходимо да го запишем в таблица на формите, за да можем да изведем изображението ѝ върху екрана.

4.2. СЪЗДАВАНЕ НА ТАБЛИЦА НА ФОРМИТЕ

Таблицата на формите е поредица от цифрови данни, включващи описанията на една или повече форми (до 255) плюс допълнителна информация за броя и местата им в таблицата (индексна част). Ще използуваме програмата МОНИТОР, за да опишем таблица на формите за двете форми, създадени в т.4.1. Данните им ще запишем от адрес $300 по следния начин:

```
* 300: 02 00 06 00 0A 00 1C 28
```

Първият байт съдържа броя на формите в таблицата. Вторият не се използува, но трябва да се остави празен, за да пази място. Следващите два байта (06 00) показват относителния адрес, от който започва описанието на първата форма спрямо началото на таблицата. Двойката байтове след тях (0A 00) определя началото на втората форма. (В компютъра Правец-82 двубайтовите стойности се записват в реда: първи младши, втори старши байт.) Следващият байт (1C) е началото на описанието на първата форма.Относителният адрес на всяка форма в таблицата се записва в отделна двойка байтове, а описанието на първата форма започва непосредствено след последния адрес. Продължаваме да въвеждаме таблицата на формите:

```
* 308: 38 00 9F 36 25 28 16 2E 24 24 35 36
       2E 24 20 15 16 25 24 00
```

Вторият байт на този ред (00) означава края на първата форма. Описанието на втората форма започва непосредствено след него с 9F и завършва с другата двойка 00.

Обърнете внимание! Всички числа, които въведохме, са шестнадесетични. Шестнадесетични са и числата, показващи

относителните адреси на разположението на формите. Поради това, че броят на формите в една таблица може да бъде значителен и числата за относителните адреси големи, когато въвеждате таблицата на формите, спазвайте следния ред. Първоначално въведете всички форми и установете действителните адреси за началото на всяка форма. Получените числа преобразувайте в шестнадесетична система и ги запишете в съответните двойки байтове за относителни адреси на всяка форма, като спазвате правилото: първи младши, втори старши байт. Например, за да въведем адрес 2085, първо го преобразуваме в $0825. Разпределяме го в два байта: младши ($25) и старши ($08). В съответната двойка за адрес на начало на форма записваме първи байт $25 и втори $08.

4.3. КОМАНДА DRAW

Ако сте изпълнили правилно описаните действия, от адрес $300 трябва да е разположена таблицата на формите, която построихме. Преди да започнем чертането на която и да е форма от нея, необходимо е да зададем началния адрес на таблицата. Той трябва да е записан в клетки 232 ($E8) и 233 ($E9) и ако пропуснем да го запишем там, не е известно какво ще се получи на екрана. Затова с помощта на програмата МОНИТОР е необходимо да въведем:

```
* E8: 00 03
```

Сега сме готови да чертаем с командата DRAW (програма 4.1).

```
1 REM * * ПРОГРАМА 4.1 * *
2 REM * * ЧЕРТАНЕ НА ФОРМА * *
10  ROT = 0
20  M = 1
30  HGR : HCOLOR = 3
40  FOR I = 40 TO  210 STEP 40
50  SCALE = M
60  DRAW 1 AT I,30
70  DRAW 2 AT I,100
80  M = M + 1
90  NEXT I
100 END
```

С ред 10 се установява нулев ъгъл на завъртане на формите (вж. т.4.5). С ред 30 се установява режим ВРС и цвят 3 (бяло I). В ред 50 на мащабния коефициент се присвоява стойността на променливата М. В процеса на изпълнението на програмата М се променя от 1 до 5. При SCALE = 1 фигурата се изчертава с размера, с който е проектирана, а при SCALE = 5 се увеличава пет пъти. SCALE може да заема стойности от 1 до 255, но мно-

го големият мащаб изкривява формата. Ако запишем SCALE = = 0, това съответствува на 256, при което формата ще се увеличи до такава степен, че няма да можем да я познаем.

С ред 60 се чертае първата форма, като се започва от точката с координати !,30. С ред 70 се чертае втората форма, като се започва от !,100. Координатите, които се задават в командата DRAW, показват мястото на точката, в която започва изпълнението на първия вектор от описанието на формата.

Ако всички разгледани операции са изпълнени правилно, с програма 4.1 ще се начертаят 5 копия на двете форми, всяко от които е по-голямо от предишното.

4.4. СЪХРАНЯВАНЕ НА ФОРМИ

Ако разполагаме с дисково запомнящо устройство, можем да запазим таблицата на формите, като запишем съдържанието на областта от паметта, която я съдържа, върху дискета чрез командата BSAVE. Обърнете внимание, че последната двойка 00 трябва да се записва винаги заедно с останалата част от таблицата. В противен случай се получават непредсказуеми резултати. За да запишем таблицата на формите, която създадохме върху дискетата, необходимо е да въведем:

```
BSAVE ТАБЛИЦА,A$300,L$1C
```

или

```
BSAVE ТАБЛИЦА,A768,L28
```

Ако към програмата 4.1 добавим редовете:

```
22  D$ = CHR$ (4): REM МК – Д/D
24  PRINT D$;"BLOAD ТАБЛИЦА"
26  POKE 232,0: POKE 233,3
```

таблицата ще се зареди автоматично в паметта от адрес $300.

Когато таблицата на формите е голяма, може да не се събере в пространството след адрес $300. Там има само 192 свободни байта ($300 – $3BF). По-дългите таблици се разполагат обикновено в горната част на паметта, като указателят HIMEM се установява да сочи първия свободен байт под тях. Целта е да се предотврати разполагането на променливите на програмата, използуваща формите, в същата област, в която те се намират. За микрокомпютъра Правец-82, който има памет 48 Кбайта, ДОС установява HIMEM на адрес $9600 (38 400). Ако таблицата, която използувате, е дълга $100 (256) байта, трябва да въведете:

```
BLOAD име на таблица,A$9500
HIMEM: 38143
```

или

```
BLOAD име на таблица,A38144
HIMEM: 38143
```

Ако искате да съхраните таблицата върху магнитна лента, можете да използувате командата WRITE от МОНИТОРА, за да запишете съдържанието на областта от паметта, която я съдържа. В този случай е необходимо да определите шестнадесетичните стойности на първия и последния адрес и дължината на таблицата. За нашата таблица тези стойности са съответно: $300, $31B и $1C. Първо запишете дължината на таблицата в клетки 0 и 1, като въведете:

```
* 0: 1C 00
```

След това запишете съдържанието на клетки 0 и 1 върху лентата и непосредствено след тях съдържанието на таблицата. За да извършите всичко това в една команда, въведете:

```
* 0.1W 300.31BW
```

Ако искате отново да заредите таблицата в паметта, можете да използувате командата READ на МОНИТОРА. Версията на БЕЙСИК за Правец-82 разполага с по-добра възможност. Командата SHLOAD зарежда таблици на форми от магнитна лента и ги поставя непосредствено под HIMEM, актуализира HIMEM да сочи адреса под таблицата и поставя началния адрес на таблицата в клетки $E8 (232) и $E9 (233). Удобството на тази команда е, че всички операции се извършват автоматично. Единствената команда, която трябва да въведете, е SHLOAD. За разлика от мониторните команди тя може да се използува от програма на БЕЙСИК. На практика, за да заредите таблица на форми чрез SHLOAD, трябва да пренавиете лентата до началото на таблицата, да натиснете бутона за възпроизвеждане (PLAY) и да въведете:

```
SHLOAD.
```

4.5. ВЪРТЕНЕ НА ФОРМИ. КОМАНДА ROT

При ROT = 0 формата се ориентира по същия начин, както е проектирана – без завъртане. Когато ROT = 16, тя се изчертава завъртяна на 90° по посока на часовниковата стрелка спрямо оригинала. При ROT = 32 тя е завъртяна на 180°, а при ROT = 48 – на 270°. Командата ROT = 64 я поставя отново в изходно положение. Така можем да продължим до ROT = 256.

Стойността, която се задава в командата ROT, зависи частично от текущата стойност на SCALE. При SCALE = 1 за ROT има 8 валидни стойности (0, 8, 16, 24, 32, 40, 48 и 56), а при SCALE = 2 и 3 – съответно 16 и 32. По този начин при по-голям мащаб възможните стойности на завъртане се увеличават. Ако по някаква причина в командата ROT зададем невалидна стойност, формата се завърта до положението, което съответствува на следващата по големина валидна стойност. Тези особености се демонстрират от програмата 4.2. За да я изпъл-

ните, необходимо е да разполагате с дискета, върху която е записана (чрез BSAVE) таблицата на формите, която създадохме.

```
 1 REM **ПРОГРАМА 4.2**
 2 REM **ВЪРТЕНЕ НА ФОРМА**
 10 REM *ЗАРЕЖДАНЕ НА АДРЕСА НА ТАБЛИЦАТА*
 20 POKE 232,0: POKE 233,3
 30 REM *ЗАРЕЖДАНЕ НА ТАБЛИЦАТА*
 40 PRINT  CHR$ (4);"BLOAD ТАБЛИЦА,A$300"
 50 REM *ВЪВЕЖДАНЕ НА МАЩАБ*
 60 HOME : VTAB 21: HGR : HCOLOR = 3
 70 INPUT "МАЩАБ?";M$: SCALE = VAL(M$)
 80 R =0
 90 REM *ЧЕРТАНЕ*
100 FOR I = 10 TO 244 STEP 26
110 ROT = R
120 DRAW 2 AT I,100
140 PRINT " ";R;" ";
150 R = R + 2
160 NEXT I
175 VTAB 23: HTAB 1
180 PRINT "СТОЙНОСТИ НА ЗАВЪРТАНЕ"
190 VTAB 24: PRINT "ЗА ДА ПРОДЪЛЖИТЕ – С/Ц":
    GET C$: IF C$ = "С" OR C$ = "Ц" GOTO 60
200 END
```

След като стартирате програмата, наблюдавайте степените на завъртане на фигурите и стойностите, които се появяват под тях. За да прекъснете програмата, натиснете който и да е клавиш освен Ц/С.

4.6. КОМАНДА XDRAW

Да предположим, че с командата DRAW сме начертали форма, чийто цвят се различава от цвета на полето, върху което е разположена. Ако се опитаме да я изтрием, като я начертаем отново с черен цвят, на мястото ѝ ще остане черно петно. Съществува възможност за целта да използуваме цвета на фона. Тогава възниква проблем, ако формата застъпва няколко разноцветни фигури. Повторното чертане с цвета на която и да е от тях или с цвета на фона разрушава останалите форми.

За решаването на тези проблеми е създадена командата XDRAW. Тя се използува по същия начин както DRAW. Разликата е, че вместо да чертае с установения от HCOLOR цвят, тя използува цветове, които зависят изцяло от фона. С командата XDRAW всяка точка на формата се изчертава автоматично с цвят, който е допълнителен на досегашния.

Допълващи се цветове са: бяло и черно, синьо и зелено, оранжево и виолетово. Например, ако чертаем с XDRAW върху зелено поле, ще получим синя фигура. Ако с XDRAW се засегне бяло поле, то става черно и т.н.

Ползата от XDRAW е, че ако с нея начертаем повторно дадена фигура, тя се изтрива напълно и възстановява цвета на фона. Тази характеристика дава значителни удобства при осъществяване на анимация.

4.7. РЕДАКТОР НА ФОРМИ И ТАБЛИЦИ НА ФОРМИ

В приложената дискета е включен редактор за създаване и работа с форми и таблица на форми. Той се осъществява от програмите ПР 4.3 и ПР 4.3А и използува двоичния масив МРЕЖА.

Главните функции на редактора се извеждат на екрана веднага след стартиране на програмата ПР 4.3. Те съставят основното меню, което включва :

създаване на форма;
записване на форма;
четене на форма;
дешифриране на форма;
каталог;
създаване на таблица;
записване на таблица;
четене на таблица;
край.

Потребителят избира функциите от менюто,като движи светещия маркер чрез стрелките МК-Н/Х и МК-U/Y и натиска RETURN, когато маркерът се намира срещу желаната функция.

Ще опишем действието на отделните функции.

1. **Създаване на форма.** Тази функция е основна, тъй като създаването на таблица е възможно само ако формите, които тя включва, са записани върху дискета. Веднага след избирането на функцията се проверява дали в оперативната памет има форма, която не е записана върху дискетата. Ако има такава, потребителят отговаря с Д или Н, за да потвърди или предотврати изтриването ѝ. След това редакторът избира режим ВРС и смесен екран. В графичната част на екрана се начертава мрежа с размери 25 х 25, където се поместват новосъздадените форми. Потребителят задава началните координати, от които желае да започне рисуването на формата. На съответното място в мрежата се появява мигащ маркер.

В текстовата част на екрана се извежда съобщението ЧЕРТАЕ или НЕ ЧЕРТАЕ и описание на действието на клавишите ОСВ и ?. Клавишът ОСВ връща управлението към главното меню, а ? извиква помощен екран, който съдържа описание на действие-

то на останалите клавиши, използувани от редактора. Те са следните:

с шината за интервал се избират режимите за чертане, не чертане и изтриване;

с клавишите И/I, Й/J, К/K и М/M маркерът се движи съответно нагоре, наляво, надясно и надолу;

с едновременно натискане на клавишите МК-Е/E се изтрива цялата форма;

с едновременно натискане на клавишите МК-П/P се променят началните координати.

В процеса на рисуване формата се изобразява в мрежата с увеличени размери. Заедно с това вдясно на екрана се изобразява с нормални размери.

След завършване на рисуването потребителят се връща в главното меню (с клавиша ОСВ), за да избере функция за записване. До този момент изображението на формата не е кодирано с векторите за описание на формата.

2. **Записване на форма.** При тази функция нарисуваната фигура се сканира и автоматично се генерират числовите стойности, определящи векторното описание на формата. Те се групират по две или три (ако е възможно) в един байт и се съхраняват в паметта, докато се запишат върху дискета с команда BSAVE. За целта е необходимо потребителят да зададе името на файла, в който ще се записва формата. Ако той отговори само с RETURN, формата не се записва. Тя също не се записва, ако потребителят вместо с име на файл отговори с ОСВ. Тогава се преминава направо към основното меню. При повторно извикване на функцията за запис на форма редакторът проверява дали има незаписана форма и ако има такава, на екрана се появява въпрос дали да бъде изтрита или записана. Името на файла, което потребителят въвежда, трябва да не е по-дълго от 10 знака. Редакторът добавя автоматично пред него символите „F.".

*3.***Четене на форма.** С тази функция в паметта на компютъра се зареждат форми, които са записани предварително върху дискета. Веднага след влизане във функцията редакторът предупреждава, че ако в паметта има друга форма, тя ще бъде изтрита и пита дали да продължи. След като въведе съответния отговор, потребителят задава името на файла, от който желае да зареди форма. То не трябва да съдържа добавените от редактора символи "F.". Прочетената форма се начертава върху екрана и програмата влиза автоматично във функцията създаване.

4. **Дешифриране на форма.** В текстов режим се извеждат шестнадесетичните и десетичните стойности на байтовете, съдържащи вектори на форми. Екранът се „прелиства", като се

натисне който и да е клавиш освен ОСВ, с който управлението се връща към основното меню.

***5.* Каталог.** Действието на тази функция е еднакво с действието на командата CATALOG от ДОС. Функцията дава възможност да се видят имената на файловете от използваната дискета, без да се излиза от редактора.

***6*. Създаване на таблица.** С тази функция се получава таблица на форми, които са създадени предварително и се съдържат в отделни файлове на дискетата. Потребителят задава броя на формите, които иска да включи в таблицата, и имената на файловете, които ги съдържат. При създаването на таблицата всяка форма се извежда на екрана и може да бъде отхвърлена, преди да бъде включена в нея.

***7*. Записване на таблица**. Тази функция се изпълнява след създаване на таблицата с предишната функция. Потребителят задава името на файла, в който се записва таблицата. То трябва да не е по-дълго от 10 знака. Редакторът поставя автоматично пред него символите "Т.". По този начин имената на файловете, съдържащи таблици на форми, се различават от имената на файловете, съдържащи форми.

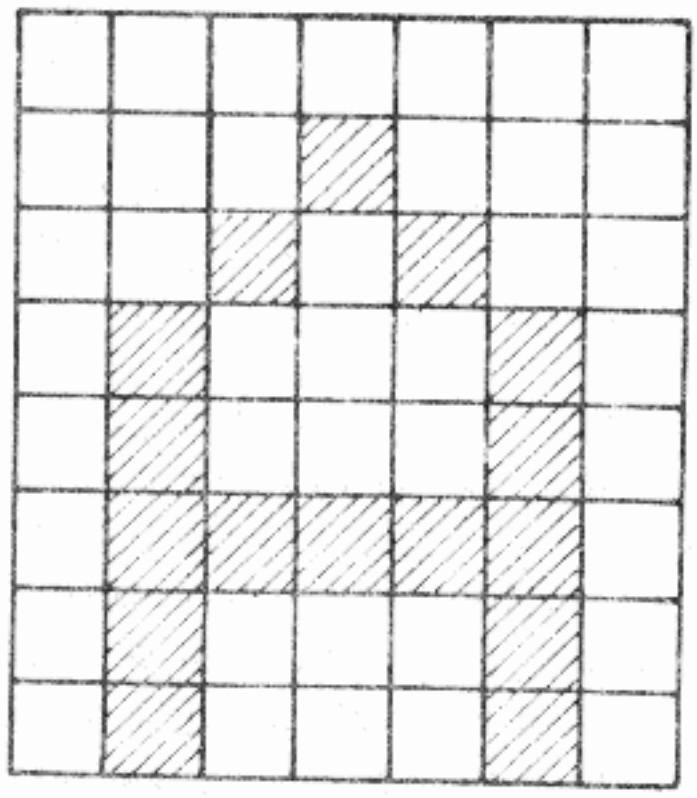
***8*.Четене на таблици**. С тази функция се зареждат в паметта таблици на форми, записани върху дискета. Функцията позволява да се изведат последователно на екрана всички форми, които се съдържат в заредената таблица.

ГЛАВА 5. ТЕКСТ В ГРАФИЧЕН РЕЖИМ

Версията на БЕЙСИК за микрокомпютъра Правец-82 няма достатъчно развити възможности за непосредствено изобразяване на текст в графичен режим. Възможно е да се извеждат четири реда текст в най-долната част на екрана, но това е незадоволително. За да изпълняват пълноценно предназначението си, надписите трябва да се поставят близо до графичните фигури, които пояснявят. В тази глава ще разгледаме два основни метода за изобразяване на текст в графичен режим с висока разделителна способност. Те са пояснени с примерни програми, които лесно могат да се допълнят, за да удовлетворят конкретните изисквания на различни приложения.

5.1. ИЗОБРАЗЯВАНЕ НА ТЕКСТ ЧРЕЗ БИТОВИ КОМБИНАЦИИ

Всеки знак в нормален текстов режим се изобразява чрез подходяща комбинация от светли и тъмни точки, които се помещават в правоъгълна матрица с размери 7 × 8 точки. Увеличеното изображение на буквата А, осъществено по този начин, е показано на фиг. 5.1. От друга страна, графичният екран за режим ВРС се състои от 192 реда по 280 точки. Ако го разделим на правоъгълни матрици с размери 7 × 8, ще получим 24 реда по 40 блока. Сравнението на тези стойности с размерите на знаците в текстов режим ни подсказва как можем да изобразим текст в графичен режим.



фиг. 5.1

За да изобразим числото 5 в най-горния ляв ъгъл на екрана, необходимо е да определим битовите комбинации, които трябва да съдържат седемте байта, управляващи изображението на съответното правоъгълниче. Адресите на байтовете в лявата колона на табл. 5.1 са взети от картата на паметта за режим ВРС.

Останалата част от табл. 5.1 показва комбинациите от светли и тъмни точки (точков модел), изобразяващи символа 5 и съответствуващите им двоични и десетични стойности във всеки байт.

Таблица 5.1

Адрес	Точков модел	Стойност	
		Двоична	Десетична
8192	– – – – – – –	00000000	0
9216	– × × × × × –	00111110	62
10240	– × – – – – –	00000010	2
11264	– × × × × – –	00011110	30
12288	– – – – – × –	00100000	32
13312	– – – – – × –	00100000	32
14336	– × – – – × –	00100010	34
15360	– – × × × – –	00011100	28

Следователно числото 5 можем да изведем върху екрана със следната последователност от команди:

```
HGR
POKE 8192,0: POKE 9216,62:
POKE 10240,2: POKE 11264,30:
POKE 12288,32: POKE 13312,32:
POKE 14336,34: POKE 15360,28
```

По аналогичен начин можем да изведем всеки друг символ. Първо преобразуваме битовите комбинации, които го изобразяват, в двоични и десетични стойности. След това ги записваме в подходящи адреси от паметта. Най-просто се адресират символите, които се вместват в правоъгълна матрица с размери 7 × 8 точки. По-големите символи могат да се изобразят чрез комбинация от няколко блока с размери 7 × 8 точки.

Въпреки че принципно изяснихме извеждането на символи върху графичен екран, необходимо е да разполагаме с прости начини за достъп до битовите комбинации, изобразяващи отделните символи. Освен това ни е необходим алгоритъм за определяне на мястото от екрана, върху което искаме да ги разположим.

Един от начините да решим първия проблем е да използуваме двумерен масив. Тъй като общият брой кодове за символи в Правец-82 е 96 и всеки от символите се изобразява с 8 комбинации от битове, размерността на масива трябва да е (95,7). Ако

номерираме символите в масива SM%(95,7) по такъв начин, че знакът 5 се намира на позиция 21, за елементите на масива, чийто първи индекс е 21, трябва да имаме:

```
SM%(21,0) = 0
SM%(21,1) = 62
SM%(21,2) = 2
SM%(21,3) = 30
SM%(21,4) = 32
SM%(21,5) = 32
SM%(21,6) = 34
SM%(21,7) = 28
```

Тогава символът 5 можем да изобразим в горния ляв ъгъл на екрана, като изпълним следната програма:

```
10  HGR
20  FOR I = 0 TO 7
30  ADR = 8192 + I * 1024
40  POKE ADR,SM%(21,I)
50  NEXT I
```

Първите 32 кода на таблицата от приложение 1 не съответстват на символи, които можем да изобразим на екрана. Затова, ако дадена символна променлива, например CH$, съдържа някакъв символ, удобно е номерата на елементите от масива, съдържащи битовите комбинации на този символ, да определим чрез: SM%(N,0), SM%(N,1),...,SM%(N,7), където

N = ASC(CH$) − 32

За да решим как да определяме мястото от екрана, върху което се извеждат символите, необходимо е да разгледаме още веднъж картата на паметта за режим BPC (фиг. 3.1). От нея се вижда, че адресите на осемте байта, управляващи изображението на правоъгълничето, намиращо се най-горе вляво на екрана, се променят от 8192 до 15360 през 1024. Непосредствено под реда с адрес 15360 започва ново правоъгълниче с първоначален адрес 8320. Адресите в него се променят от 8320 до 15488 също през 1024. Разликата между двата основни адреса 8320 и 8192 е 128. По същия начин се променят адресите на байтовете, управляващи изображението на правоъгълничетата от първите 8 реда. Те образуват горната третина на екрана. В нея основният (базовият) адрес OA на всеки ред се определя по формулата:

OA = 8192 + 128 * I,

където I се променя от 0 до 7.

Вътре във всяко правоъгълниче осемте байта се адресират чрез израза:

OA + 1024 * J,

където J се променя от 0 до 7.

Средната третина от екрана се адресира по същия начин, като за първоначален адрес вместо 8192 се използува 8232 = 8192 + 40. Най-долната третина на екрана се подчинява на същото правило, но има първоначален адрес 8272 = 8232 + 40.

Следователно, за да определим адреса на байта, чието съдържание искаме да изведем в най-лявата колона от екрана, необходимо е да знаем:

в коя третина от екрана се намира редът, в началото на който искаме да изведем символа;

номера на реда в съответната третина;

номера на байта в правоъгълничето.

Тъй като при извеждане на текст обикновено се работи с цели правоъгълничета, по-важни са първите две точки.

За удобство горната, средната и долната третина на екрана ще номерираме съответно с 0, 1 и 2. Освен това ще номерираме с цифрите от 0 до 7 (отгоре надолу) осемте реда правоъгълничета във всяка третина. Ще използуваме също променливите HT и VT (производни на HTAB и VTAB), за да означаваме мястото от екрана, върху което трябва да се разположи текстът. Тогава VT = 1÷8 определя редовете в третина 0; VT = 9÷16 – в третина 1; VT = 17÷24 – в третина 2.

В резултат номерът на третината NT можем да определим чрез

NT = INT((VT−1)/8).

Следващата задача е да определим номера на реда NR вътре в третината. Зависимостите между величините VT, NT и NR са показани в табл. 5.2. Като използуваме тези зависимости, можем да определим NR по формулата

NR = VT−1−8*INT((VT−1)/8).

Основния адрес 0A за всеки ред правоъгълничета можем да изразим чрез:

OA = 8192+40*NT+128*NR,

а като заместим NT и NR, ще получим:

OA = 8192+40*INT((VT−1)/8)+128*(VT−1−8*INT((VT−−1)/8)).

След като сме в състояние да определим основния адрес на реда в зависимост от VT, необходимо е да отчетем и променливата HT, за да определим основния адрес на всяко правоъгълниче вътре в реда. Това не е трудно, тъй като основните адреси на съседните правоъгълничета по продължение на един ред се различават само с 1. В резултат за основния адрес на правоъгълничето, зададено чрез VT и HT, получаваме

Таблица 5.2

VT	NT	NR	VT	NT	NR	VT	NT	NR
1	0	0	9	1	0	17	2	0
2	0	1	10	1	1	18	2	1
3	0	2	11	1	2	19	2	2
4	0	3	12	1	3	20	2	3
5	0	4	13	1	4	21	2	4
6	0	5	14	1	5	22	2	5
7	0	6	15	1	6	23	2	6
8	0	7	16	1	7	24	2	7

$$OA = 8192 + 40*INT((VT-1)/8) + 128*(VT-1-8*INT((VT-1)/8)) + HT-1.$$

След като опростим, окончателната формула за основния адрес е

$$OA = 8063 + 128*VT - 984*INT((VT-1)/8) + HT.$$

Необходимо е още да осигурим лесен достъп до битовите комбинации, изобразяващи отделните знаци. Разполагането им в масив изисква последният да е постоянно на разположение на програмата, която ги извежда. От друга страна, ръчното му зареждане с битовите комбинации за всички символи е доста трудна работа. Поради това не е рационално той да се вписва винаги в програмите, извеждащи текст в графичен режим. Съхраняването на масива като текстов файл също е неудобно. То изисква значителна дискова памет и е свързано със сравнително бавен обмен на информацията между програмите и файла. Затова ще разгледаме един по-различен подход, който на пръв поглед изглежда ненужно усложнен, но е достатъчно бърз и икономичен. Той е осъществен практически в програмите 5.1 и 5.2.

```
  1 REM **ПРОГРАМА 5.1**
  2 REM **СЪХРАНЯВАНЕ НА СИМВОЛНИЯ МАСИВ**
 10 ADR = 0:DL = 0
 20 DIM SM%(95,7)
 25 REM *ЗАРЕЖДАНЕ НА МАСИВА*
 30 FOR I = 0 TO 95
 40 FOR J = 0 TO 7
 50 READ SM%(I,J)
 60 NEXT J,I
 65 REM *ЗАПИС ВЪРХУ ДИСКЕТА*
 70 ADR = PEEK (107) + 256 * PEEK (108)
 80 DL = PEEK (109) + 256 * PEEK (110) - ADR
 90 PRINT CHR$ (4);"BSAVE СИМВОЛИ,A";ADR;",L";DL
100 REM *ЧИСЛОВИ СТОЙНОСТИ ЗА ИЗОБРАЖЕНИЯ НА
    СИМВОЛИ*
```

```
105  REM ИНТЕРВАЛ
110  DATA 0,0,0,0,0,0,0,0
115  REM  !
120  DATA 0,8,8,8,8,0,8
125  REM  "
130  DATA  0,20,20,20,0,0,0,0
.
.
.
1035  REM ]
1040  DATA 0,62,48,48,48,48,48,62
1045  REM `
1050  DATA 0,0,8,20,34,0,0,0
1055  REM    DEL
1060  DATA 0,0,0,0,0,0,0,0
1065  END
```

С първата от тях (програма 5.1) се зарежда двумерният масив SM%(I,J) и се записва съдържанието му като двоичен файл върху дискета. Индексът I се променя от 0 до 95. Всяка негова стойност съответствува на един от символите, които искаме да изобразим. Индексът J се променя от 0 до 7. Неговите стойности съответствуват на редовете от правоъгълничето, в което се изобразява символът с номер I. Номерът на символа, съдържащ се в променливата CH$, се определя съгласно зависимостта

I = ASC(CH$) − 32.

Числовите стойности, задаващи изображението на отделните символи, са записани в команди DATA след ред 100. Те са получени в резултат на предварителното изчертаване на символите в матрица с размер 7 × 8 точки. От тези фигури са определени двоичните стойности на всеки байт, управляващ отделен ред от матрицата. Те на свой ред са преобразувани в десетични стойности. Тук в текста на програмата 5.1 не са показани всички редове, съдържащи числови стойности за изображения на символи. Пълната програма е представена в приложената към книгата дискета.

При записването на масива SM%(I,J) върху дискета се използува съдържанието на клетки 107,108,109,110. Първите две съдържат началния адрес на масива, а вторите — крайния адрес (младшите байтове се записват първи). На тази основа се определят (редове 70,80) началният адрес ADR и дължината DL на областта от паметта, чието съдържание се записва. В резултат от изпълнението на програмата върху дискетата получаваме двоичния файл с име СИМВОЛИ.

Програмата 5.2 демонстрира как се използува този файл. Тя го прочита от дискетата и зарежда съдържанието му обратно в паметта на микрокомпютъра, като възстановява масива

SM%(I,J). Отново се използуват клетки 107,108,109 и 110. Съдържанието на първите две се използува, за да се определи началото на областта, в която се разполага масивът (ред 20). Вторите две се зареждат със

```
  1  REM  **ПРОГРАМА  5.2**
  2  REM **ИЗВЕЖДАНЕ НА СИМВОЛИ ЧРЕЗ БИТОВИ КОМ
     БИНАЦИИ**
 10 ADR = 0:DL = 0
 20 ADR = PEEK (107) + 256 *  PEEK (108)
 30  PRINT  CHR$ (4);"BLOAD СИМВОЛИ,A";ADR
 40 DL = 256 *  PEEK (43617) +  PEEK (43616)
 50  POKE 110, INT ((ADR + DL) / 256)
 60  POKE 109,ADR + DL - 256 *  INT
     ((ADR + DL) / 256)
 70  HGR : HOME : VTAB 21
 80 VT = 3:HT = 8:SP$ = "ТЕКСТ В ГРАФИЧЕН РЕЖИМ"
 90  GOSUB 210
100 VT = 5:HT = 8:SP$ = "ЧРЕЗ БИТОВИ КОМБИНАЦИИ"
110  GOSUB 210
120 VT = 10:HT = 2:SP$ = "АКО ЖЕЛАЕТЕ,
    ОПИТАЙТЕ ДА ИЗВЕДЕТЕ ТЕКСТ"
130  GOSUB 210
140 VT = 12:HT = 2:SP$ = "ЗА ЦЕЛТА ВЪВЕДЕТЕ:"
150  GOSUB 210
160  INPUT "СИМВОЛНА ПОСЛЕДОВАТЕЛНОСТ:";SP$
170  INPUT "ВЕРТИКАЛНА КООРДИНАТА:";VT
180  INPUT "ХОРИЗОНТАЛНА КООРДИНАТА:";HT
190  GOSUB 210
200  END
205   REM *ИЗВЕЖДАЩА ПОДПРОГРАМА*
210 OA = 8063 + 128 * VT - 984 *
    INT ((VT - 1) / 8) + HT
220  FOR I = 1 TO LEN (SP$)
230 N = ASC ( MID$ (SP$,I,1)) - 32
240  FOR J = 0 TO 7
250  POKE OA + 1024 * J,SM%(N,J)
260  NEXT J
270 OA = OA + 1
280  NEXT I
290  RETURN
```

стойности, които сочат края на масива. Те се изчисляват в редове 40 – 60, като се използува съдържанието на клетки 43616 и 43617. В тях е записана дължината на файла, който е бил зареден чрез командата BLOAD.

След като зареди отново в паметта масива SM%(I,J), програмата 5.2 предава управлението на подпрограмата, започваща от ред 210, с която се извеждат символите. Тя получава от главната програма поредицата символи за извеждане на екрана и координатите на разположението им съответно чрез променливите SP$, HT и VT. Самото извеждане извършва символ по символ, които извлича (ред 230) последователно от променливата SP$ и като използува координатите HT и VT (ред 210), определя основните адреси, на които записва числовите стойности, задаващи изображението на съответния символ (ред 250).

Тази подпрограма може да се използува в програми, осъществяващи графични изображения и игри, за извеждане на надписи в определени места от графичния екран. Преди да бъде извикана, необходимо е да се заредят променливите SP$, HT и VT с желаните стойности. При определяне на координатата HT трябва да се има предвид дължината на символния низ. Тъй като извеждането на символи не преминава на нов ред при достигане на края на екрана, остатъкът от надписа извън тези граници може да се появи на нежелано място. Препоръчително е също да се прави проверка на условията:

$1 <= HT <= 40$ и $1 <= VT <= 24$.

Програмата 5.2 за извеждане на символи върху графичен екран може да се разшири с множество допълнителни възможности и подобрения. Например съдържащата се в нея подпрограма може да се промени така, че да извежда символите вертикално. За целта трябва да се извършат следните промени:

```
210  FOR I = 1 TO LEN(SP$)
220  OA = 8063 + 128 * VT - 984 * INT((VT - 1)/8) + HT
230  N = ASC(MID$(SP$,I,1)) - 32
240  FOR J = 0 TO 7
250  POKE OA + 1024 * J,SM%(N,J)
260  NEXT J
270  VT = VT + 1
280  NEXT I
290  RETURN
```

Ако искаме да имаме възможност едновременно за хоризонтално и вертикално извеждане на символи, можем да включим двете подпрограми в една главна програма.

Обратното печатане на символите можем да получим чрез промяната:

```
250  POKE OA + 1024 * J,SM%(N,7 - J)
```

За да осъществим автоматично преминаване на нов ред, когато се достигне краят на екрана, в последния вариант на извеждащата подпрограма трябва да променим ред 270 по следния начин:

```
270 HT = HT + 1:IF HT = 41 THEN HT = 1:VT = VT + 1
```

Разкъсването на думите при пренасянето им на нов ред можем да избегнем, като включим допълнителен цикъл, с който се броят предварително знаците на всяка дума (между два интервала) преди извеждането ѝ на екрана. Ако дължината ѝ е по-голяма от (40 – HT), извеждането преминава автоматично на нов ред.

Програма 5.2 не позволява въвеждането на някои символи (например ",") от клавиатурата. Обаче включването им в символен низ, като например

SP$ = "A,B" позволява безпрепятствената им обработка.

5.2. ИЗОБРАЗЯВАНЕ НА ТЕКСТ ЧРЕЗ ТАБЛИЦИ НА ФОРМИ

Таблиците на форми дават друга възможност за извеждане на текст върху графичен екран. За сметка на по-голяма предварителна подготовка те предоставят по-гъвкави средства за изписване и изтриване на символи.

В случая таблицата на формите съдържа предварително съставени графични изображения на символи, които могат да се разполагат бързо и лесно на различни места от екрана. Формите се идентифицират с номера. За да изведем определена форма, необходимо е да знаем номера ѝ в таблицата N и мястото на екрана (X,Y), където се разполага. Тогава използуваме или командата DRAW N AT X,Y, или XDRAW N AT X,Y.

Основните усилия при изобразяването на символи по този начин не са свързани с чертането на знаците, а с изготвянето на таблицата на формите. За да се облекчи тази задача, могат да се използуват успешно съществуващите програми – генератори на форми. Такъв например е описаният в гл.4 редактор на форми.

Програмите 5.3 и 5.4 осъществяват стандартния шрифт на Правец-82. Номерата на формите на символите се свързват с кодовете на съответните знаци, съдържащи се в променливата CH$, чрез познатата зависимост:

```
N = ASC(CH$) – 32
```

Програмата 5.3 съдържа цялата информация, необходима за съставяне на таблицата на формите, и я създава върху дискета с името ТАБЛИЦА НА СИМВОЛИ. Тя включва всички символи, които използува Правец-82.

```
1   REM **ПРОГРАМА 5.3**
2   REM **СЪЗДАВАНЕ НА СИМВОЛНАТА ТАБЛИЦА**
5  REM *ЗАРЕЖДАНЕ НА ТАБЛИЦАТА В ПАМЕТТА*
10  FOR I = 24576 TO 25819
20  READ X
```

```
30   POKE I,X
40   NEXT I
45   REM *ЗАПИС ВЪРХУ ДИСКЕТА*
50   PRINT    CHR$ (4);"BSAVE ТАБЛИЦА НА СИМВО
     ЛИ,A24576,L1243"
90   REM *ИНДЕКСНА ЧАСТ НА ТАБЛИЦАТА*
100  DATA 95,0,192,0,198,0,206,0
110  DATA 221,0,236,0,249,0,8,1
.
.
.
162  DATA 181,4,195,4,207,4,215,4
190   REM *ФОРМИ НА СИМВОЛИ*
200  REM !
205  DATA 73,4,32,36,36,0
210  REM "
215  DATA 9,64,24,32,108,54,4,0
.
.
.
1140  REM ]
1145  DATA 41,45,37,39,37,39,37,39,253,63,4,0
1150 REM ^
1155  DATA 64,24,97,12,21,21,4,0
1160  REM DEL
1165  DATA 1,0
1170  END
```

С редове 10 – 40 таблицата на формите се зарежда в паметта между адреси 24576 и 25819, а с ред 50 се записва върху дискета. Редове 100 – 162 съдържат индексната част на таблицата, докато редове 200 – 1165 съдържат числовите описания на формите на символите. В текста на програмата 5.3 не са показани всички редове, съдържащи индексната част на таблицата и описанията на формите. Пълната програма е представена в приложената към книгата дискета.

С програмата 5.4 таблицата на формите се чете от дискетата и се зарежда в паметта на микрокомпютъра. С ред 20 в клетки 232 и 233 се записва адресът, от който се разполага таблицата. Адресът се взема от клетки 43634 и 43633, съдържащи началния адрес, от който е заредила информация последната команда BLOAD.

```
1    REM **ПРОГРАМА 5.4**
2  REM **ИЗВЕЖДАНЕ НА ТЕКСТ ЧРЕЗ ТАБЛИЦИ НА ФОР
     МИ**
10    PRINT    CHR$ (4);"BLOAD ТАБЛИЦА НА СИМВО
```

```
    ЛИ,A24576"
20  POKE 232, PEEK (43634): POKE 233, PEEK (43635)
30  HOME : HGR : HCOLOR = 3: SCALE = 1: ROT = 0: VTAB 22
40 VT = 3:HT = 8:SP$ = "ТЕКСТ В ГРАФИЧЕН РЕЖИМ"
50  GOSUB 170
60 VT = 5:HT = 8:SP$ = "ЧРЕЗ ТАБЛИЦИ НА ФОРМИ"
70  GOSUB 170
80 VT = 10:HT = 2:SP$ = "АКО ЖЕЛАЕТЕ,
    ОПИТАЙТЕ ДА ИЗВЕДЕТЕ ТЕКСТ"
90  GOSUB 170
100 VT = 12:HT = 5:SP$ = "ЗА ЦЕЛТА ВЪВЕДЕТЕ:"
110  GOSUB 170
120  INPUT "СИМВОЛНА ПОСЛЕДОВАТЕЛНОСТ: "SP$
130  INPUT "ВЕРТИКАЛНА КООРДИНАТА: ";VT
140  INPUT "ХОРИЗОНТАЛНА КООРДИНАТА: ";HT
150  GOSUB 170
160  END
170 HT = 7 * (HT - 1):VT = 8 * VT - 1
180  FOR I = 1 TO  LEN (SP$)
190 N =  ASC ( MID$ (SP$,I,1)) - 32
200  IF N = 0 THEN 220
210  XDRAW N AT HT,VT
220 HT = HT + 7
230  NEXT I
240  RETURN
```

Клетки 232 и 233 съдържат съответно младшия и старшия байт на адреса на таблицата на формите. Например таблица, която е разположена в адрес A, ще стане достъпна за ползуване, ако заредим клетки 232 и 233 съответно с:

*POKE 232,A−256*INT(A/256) и POKE 233,INT(A/256).*

Ред 30 установява мащабния коефициент в 1 и коефициента на завъртане в 0. Останалата част от програмата има подобна структура на програмата 5.2. По същия начин трябва да зададем поредицата от символи, които ще извеждаме върху екрана, и мястото, където трябва да ги разположим. Тези данни се записват съответно в променливите SP$, HT и VT. Чрез тях те се предават на подпрограмата, започваща от ред 170. Последната осъществява действителното извеждане на символите, като изчислява номерата на формите в зависимост от кодовете на съответните знаци. Мястото им върху екрана се определя чрез HT и VT по същия начин както в текстов режим чрез HTAB и VTAB. Поради това е възможно да се изведат до 24 реда по 40 знака. От това следва ограничението, че не можем да извеждаме текст във всяка точка на екрана. Ако премахнем ред 180 от програмата,ще може да задаваме VT и HT в границите 0<=VT<=191 и 0<=HT<=279. Тогава ще сме в състояние да

позиционираме надписите върху екрана с точност до една точка.

Началото на формите, създавани от програмата 5.3, се намира в най-лявата долна точка от правоъгълничето, в което те се разполагат. Така че определянето на мястото на формата върху екрана се свежда до определяне на мястото на тази точка. Формата се позиционира надясно и нагоре от нея. Поради това, когато разполагаме символ в края на екрана, част от него може да премине от другата му страна. Например командата:

XDRAW 21 AT 276,16 определя точката (276,16) като долен ляв ъгъл на правоъгълната матрица, върху която трябва да се разположи символът. До края на екрана в случая няма достатъчно място, за да се побере цялата матрица, и последната колона на формата ще се начертае в лявата му част. Аналогично част от формите, разполагани близо до горния край на екрана, могат да се появят в долната му част.

Изобразяването на символи чрез таблици на форми дава възможност да се извеждат знаци с различни размери. Това се постига чрез командата SCALE. Разбира се, извеждането на букви с мащабен коефициент, по-голям от 1, изисква да се преизчислят допълнително позициите от екрана, върху които те ще се разположат. Използуването на таблици на форми позволява също да се извеждат символи, завъртени под някакъв ъгъл. Ако в програмата 5.4 зададем ROT = 16, ROT = 32 или ROT = 48, знаците ще се извеждат завъртени по посока на часовниковата стрелка съответно на 90,180 и 270°. Последователното им извеждане във вертикална посока се постига, като променим ред 220 по следния начин:

220 VT = VT + 8

Таблиците на формите предлагат още едно значително предимство пред извеждането на символи чрез битови комбинации. Изведените символи могат да се изтрият лесно, без да се наруши останалата част от графичното изображение. Това позволява да се осъществява анимация на надписите върху екрана.

ГЛАВА 6 **АНИМАЦИЯ**

Досега се занимавахме с изобразяване на статични фигури върху екрана на микрокомпютъра. Голяма част от микрокомпютърните игри осъществяват движение на фигури. За да се подготвим за тяхното програмиране, в тази глава ще разгледаме методите за анимация на прости фигури. Както при анимацията в киното, така и тук, в действителност ние ще извеждаме последователност от статични изображения, които при подходяща скорост на редуване създават впечатление за движение.

Методите за анимация на фигури се обособяват в три основни групи:

анимация чрез използуване на командата HPLOT;
анимация чрез използуване на форми и таблици на форми;
анимация чрез преместване на данни в паметта.

Всеки от методите има разновидности, а някои се препокриват частично. При избора им трябва да се има предвид, че успешното използуване на всеки от тях зависи до голяма степен от конкретните условия и целите, които трябва да се постигнат. Често се прилагат комбинирано няколко метода, за да се повиши качеството на получените изображения. Тези обстоятелства налагат да се познават разновидностите на различните методи за анимация и да се избира най-подходящият за конкретния случай.

6.1. АНИМАЦИЯ ЧРЕЗ HPLOT

Програмата 6.1 демонстрира прост случай на анимация чрез използуване на командата HPLOT. В резултат от изпълнението ѝ на екрана се появява подскачаща „топка" между „под" и „стени". Тя се представя с четири точки, които се изчертават и изтриват последователно на различни места от екрана. Движението на „топката" продължава в една посока, докато се стигне до някоя от предварително начертаните граници на „пода" или „стените". Тъй като в случая изображението е просто (само четири точки), необходимите изчисления за управлението му са минимални. Поради това движението на топката е плавно.

```
 1  REM **ПРОГРАМА 6.1**
 2  REM **ПОДСКАЧАЩА ТОПКА**
10  XC = 4:X = 8:Y = 0:A = 2:BC = 0
20  HOME : HGR
```

```
30  REM "ЧЕРТАНЕ НА СТЕНИ И ПОД"
40  HCOLOR = 2
50  HPLOT 2,0 TO 2,134 TO 266,134 TO 266,0
55  REM "ЧЕРТАНЕ НА ТОПКА"
60  HCOLOR = 3
70  HPLOT X,Y: HPLOT X + 1,Y
80  HPLOT X,Y + 1: HPLOT X + 1,Y + 1
90  IF Y = 132 THEN BC =  - BC: IF
    BC = 0 THEN BC =  - 20
100  IF X > 263 THEN XC =  - XC
110  IF X < 6 THEN XC =  - XC
120  BC = BC + A
130  HCOLOR = 0: REM  ИЗТРИВАНЕ
140  HPLOT X,Y: HPLOT X + 1,Y
150  HPLOT X,Y + 1: HPLOT X + 1,Y + 1
160  Y = Y + BC:X = X + XC
170  GOTO 60
```

В ред 50 от програмата се начертават „стените" и „подът". Следващите редове (70 и 80) чертаят „топката". Движението и в хоризонтална посока се определя от постоянната хоризонтална скорост XC. Вертикалното движение се определя от вертикалната скорост BC, която се променя с ускорение A. След като стои известно време на екрана, „топката" се изтрива с редове 140 и 150. Координатите ѝ се променят (ред 160) и тя се изчертава отново. „Отскачането" ѝ от „пода" и „стените" се управлява от редове 90 – 110. Ред 90 актуализира също и вертикалната скорост в моментите, когато Y = 132 или BC = 0. Изпълнението на програмата се прекъсва чрез МК–Ц/С или RST. Командата HPLOT се използува за изобразяване на движещи се фигури и в програма 1.6, която вече разгледахме в гл. 1. Тя се различава от предишната по това, че движещите се фигури са по-сложни. Необходими са повече изчисления, за да се определят координатите им, и съответно повече време. Поради това, ако изчертаването и изтриването им се върши направо върху екрана, то става забележимо. За да се избегне този недостатък, се използуват двете графични страници. Съдържанието на едната се извежда върху екрана, а в другата се чертае нова фигура или се изтрива старата. В същото време се изчисляват координатите, определящи положението на новата фигура.

При по-сложни изображения тези изчисления изискват значително време, което забавя извеждането. Един начин да се ускорят изчисленията е да се програмира на машинен език. Той изисква значително повече усилия и не се разглежда в тази книга. Друг изход е да се използува компилатор за БЕЙСИК, с който програмата да се преведе предварително на машинен език, а след това само да се изпълнява. Съществуват случаи, когато

изчисленията са толкова големи, че и този метод не помага. Тогава изчисленията се извършват предварително и се запомнят резултатите. След това те се използуват непосредствено при осъществяване на чертането.

```
 1  REM **ПРОГРАМА 6.2**
 2  REM **ВЪРТЯЩ СЕ КВАДРАТ**
10  DR = 3.14159 / 20
20  DIM X(60),Y(60)
30  FOR I = 1 TO 5
40  READ OX(I),OY(I)
50  NEXT I
55  REM *ПРЕДВАРИТЕЛНО ИЗЧИСЛЕНИЕ*
60  FOR I = 0 TO 9
70  R = I * DR
80  FOR J = 1 TO 5
90  X(5 * I + J) = COS (R) * OX(J) - SIN (R) * OY(J) + 140
100 Y(5 * I + J) = SIN (R) * OX(J) + COS (R) * OY(J) - 96
110  NEXT J
120  NEXT I
130  FOR I = 51 TO 60
140  X(I) = X(I - 50):Y(I) = Y( - 50)
150  NEXT I
155  REM *ИЗВЕЖДАНЕ*
160  HGR : HGR2 :I = 1
170  HCOLOR = 0: POKE 230,32: GOSUB 300
180  I = I + 10: HCOLOR = 3: GOSUB 300
190  POKE  -16300,0: POKE 230,64 :I = I - 5: HCOLOR = 0:
     GOSUB 300
200  I = I + 10: HCOLOR = 3: GOSUB 300
210  POKE  -16299,0: I = I - 5: IF I = 51 THEN I = 1
220  GOTO 170
300  HPLOT X(I),Y(I) TO X(I + 1), Y(I + 1) TO X(I + 2),Y(I +
     2) TO X(I + 3),Y(I + 3) TO X(I + 4) ,Y(I + 4): RETURN
400  DATA 80,0,0,80,-80,0,0,-80,80,0
```

Този метод е използуван в програмата 6.2. Тя осъществява изображение на въртящ се квадрат. Координатите на върховете за различните положения на квадрата през време на въртенето се записват в масивите X(60) и Y(60) след предварително изчисление. Използува се един изходен (основен) квадрат, който е центриран спрямо началото на координатната система. Той не се извежда върху екрана. Координатите на върховете му са поместени в блок от данни на ред 400. Те се зареждат в масивите OX и OY чрез командата READ от ред 40. С редове 60 – 120

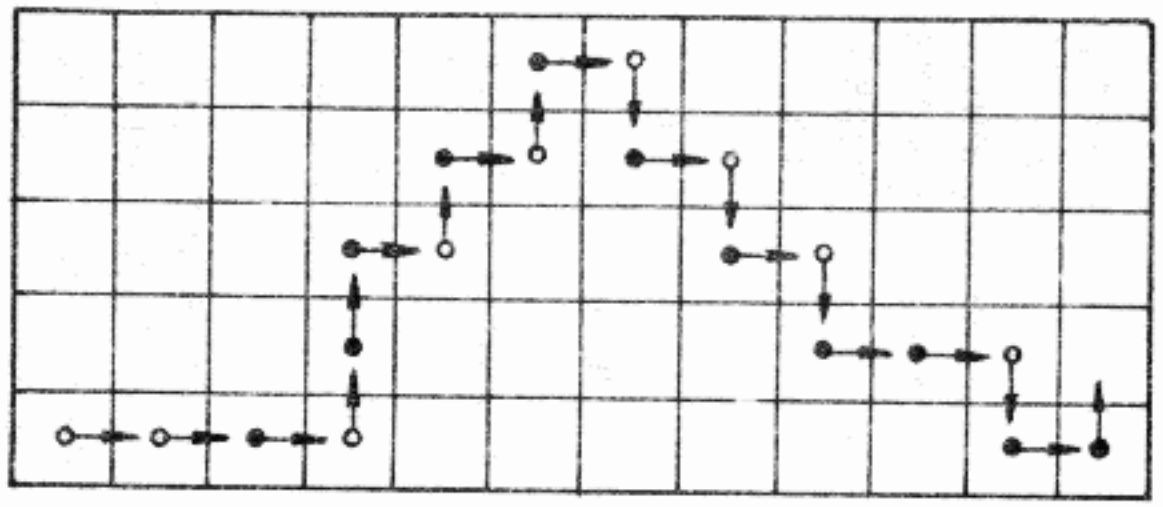
се изчисляват координатите на десет квадрата. Всеки от тях се получава чрез завъртане под ъгъл R (редове 90 – 100) на основния квадрат и преместването му до центъра на екрана. За първия квадрат ъгълът R е 0, за втория – π/20, за третия – 2π /20, за четвъртия – 3π /20 и т.н. (редове 10 и 70). Всяка двойка координати се запомня в масивите X(I) и Y(I). След като се изчислят координатите на десетте квадрата, с редове 130 – 150 данните на първите два се записват повторно от 51-ви до 60-ти елемент на масивите X(I) и Y(I). По този начин се получават още два квадрата 11 и 12, които са еднакви съответно с квадрати 1 и 2. Те са въведени, за да се улесни изчертаването. Последователността от завъртени квадрати се извежда с редове 160 – 210 и 300. С редове 160 – 180 се изтрива квадратът от първа графична страница и се начертава нов. С ред 190 на екрана се извежда първа страница, подготвя се втора за чертане и се изтрива съществуващият в нея квадрат. След това с ред 200 се чертае нов квадрат във втора страница, а с ред 210 тя се извежда на екрана и се подготвя първа страница за чертане. Управлението се изпраща отново на ред 170, за да започне изтриването на стар и подготвянето и извеждането на нов квадрат. Така процесът се повтаря непрекъснато.

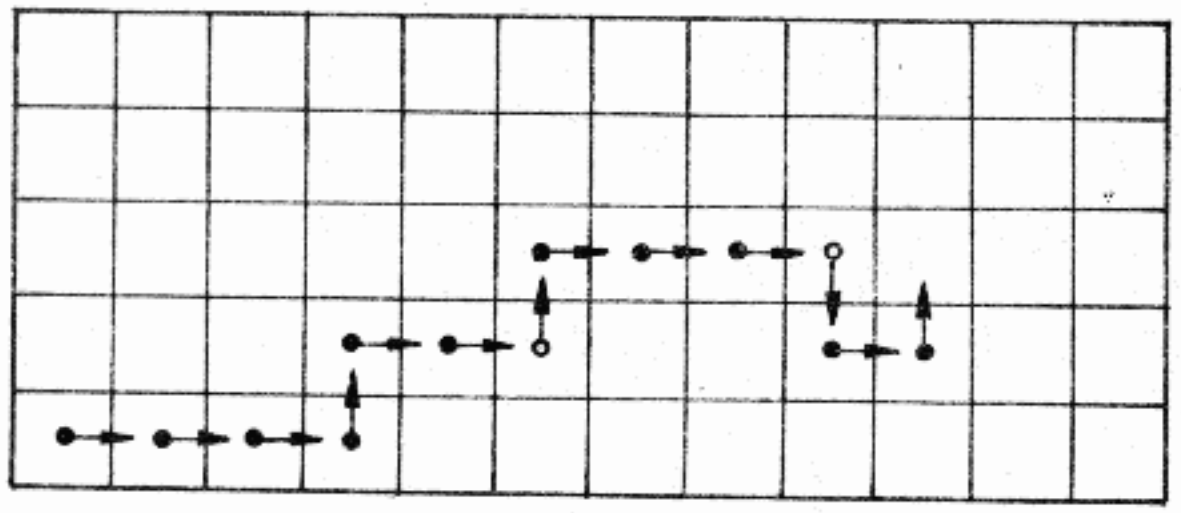
При анимацията на обектите дотук изтривахме изображенията, като ги изчертавахме допълнително с черен цвят. Другата възможност е да изтрием целия екран, например чрез CALL 62450. Често това е по-лошият вариант, защото отнема повече време (с изключение на случаите, когато трябва да се изчертава много сложно изображение). Освен това по този начин се изтриват и останалите фигури от екрана, които не са обект на анимация и трябва да останат постоянно.

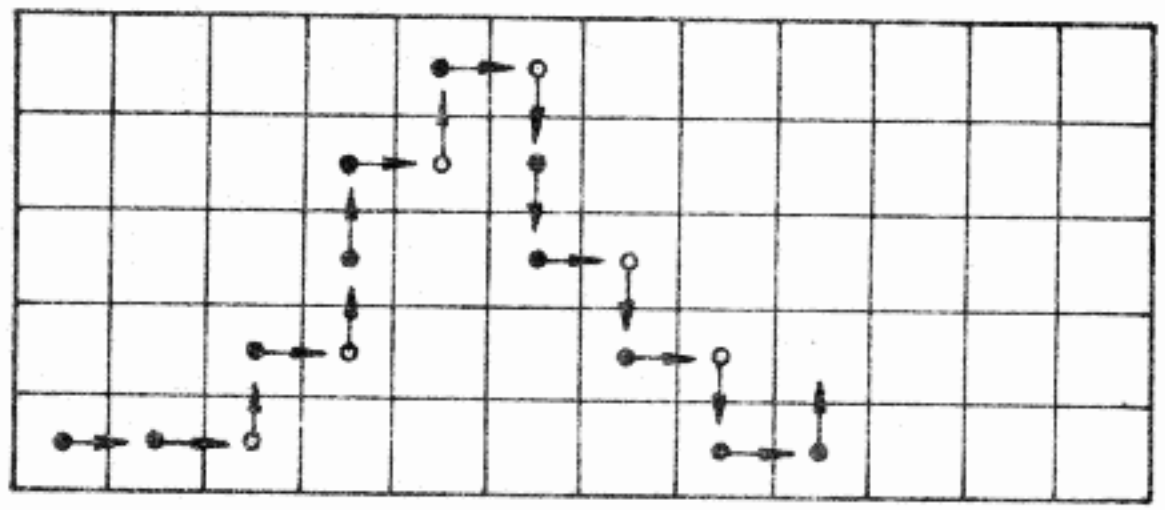
6.2. АНИМАЦИЯ ЧРЕЗ ИЗПОЛЗУВАНЕ ТАБЛИЦИ НА ФОРМИ

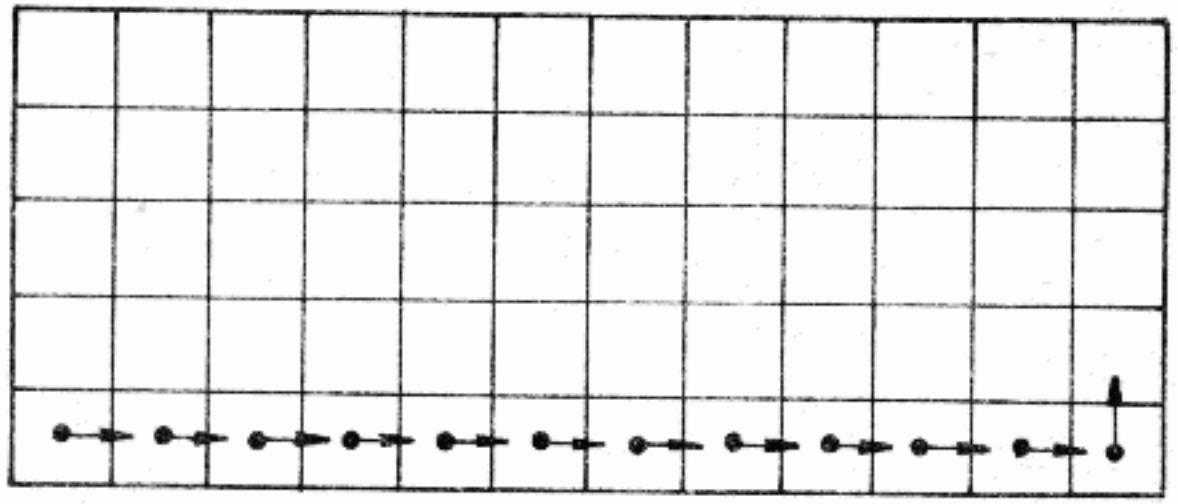
Както разбрахме, и най-простият случай на анимация чрез използуване на HPLOT изисква предварителна подготовка. Затова се предпочита да се използуват таблици на форми. Те също изискват предварителна подготовка най-малкото, за да бъдат създадени, но имат и съществени предимства.

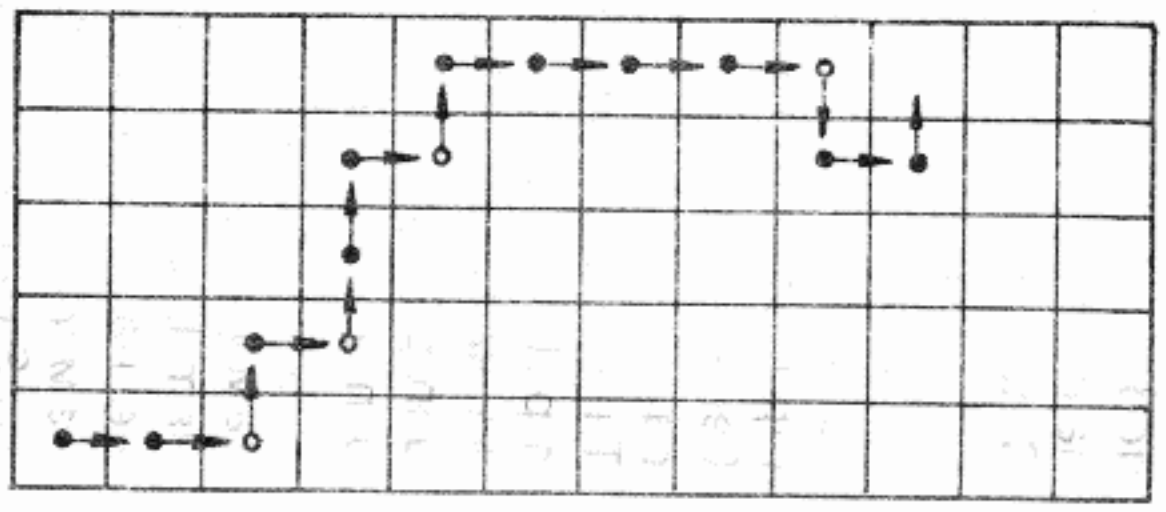
Движението в този случай се постига, като се чертаят форми последователно в различни положения на екрана. Програмата 6.3 демонстрира този метод, като имитира движение чрез чертане на последователност от различни форми. Те са създадени предварително чрез редактор на форми и отразяват пет основни положения, които заема пълзящо животно (фиг. 6.1).











фиг. 6.1

```
 1  REM **ПРОГРАМА 6.3**
 2  REM **ПЪЛЗЯЩО ЖИВОТНО**
10  FOR I = 768 TO 824
20  READ A: POKE I,A
30  NEXT I
40  POKE 232,0: POKE 233,3
50  HOME : HGR : HCOLOR = 3: SCALE = 1: ROT = 0
60  FOR X = 1 TO 270 STEP 4
70  FOR I = 1 TO 5
80  XDRAW I AT X,100
90  FOR A = 1 TO 60: NEXT A
100  XDRAW I AT X,100
110  NEXT I,X
120  TEXT : END
1000  REM *ТАБЛИЦА НА ФОРМИ*
1010  DATA 5,0,12,0,21,0,30,0,39,0,46,0
1020  DATA 45,40,32,5,168,174,21,37,0
1030  DATA 45,40,32,5,168,174,21,37,0
1040  DATA 45,5,40,5,40,45,21,37,0
1050  DATA 45,45,45,45,45,37,0
1060  DATA 9,5,32,5,40,168,21,21,173,37,0
```

Формите се задават в програмата като блок от данни в редове 1010 – 1060. Четат се с командата READ (ред 20) и се разполагат от адрес $300(768) в паметта на микрокомпютъра. С ред 40 този адрес се записва в клетки 232 и 233. С редовете от 60 до 110 последователно се чертаят и изтриват петте форми в различни точки с постепенно нарастваща хоризонтална координата. Командата XDRAW се използува за чертане и изтриване. Тя чертае формата с цветове, които допълват цветовете, намиращи се в момента върху съответните точки от екрана. Затова следващото изтриване на формата е много лесно. Тя просто се начертава повторно с XDRAW, при което се изтрива. Този подход позволява да имитира медвижение върху фона на други изображения, без да ги разрушаваме, но изисква да изтриваме и чертаем винаги цялата форма. Ако тя е по-сложна и плътно начертана, тази операция изисква значително време.

За да се избегне това неудобство, се изтриват и чертаят само онези части от формата, които действително се променят в резултат на движението.При анимациятачрез използуване на форми този метод се нарича *чертане с предварително изместване*.

За да го демонстрираме, нека в режим на програмата МОНИТОР (CALL –151) от адрес $6000 да заредим следната таблица на форми.

```
6000:02 00 06 00 45 00 3F 3F
6008:3F 3F 3F 3F 08 2D 2D 2D
6010:2D 2D 2D 18 3F 3F 3F 3F
```

```
6018:3F 3F 08 2D 2D 2D 2D 2D
6020:2D 18 3F 3F 3F 3F 3F 3F
6028:08 2D 2D 2D 2D 2D 2D 18
6030:3F 3F 3F 3F 3F 3F 08 2D
6038:2D 2D 2D 2D 2D 18 3F 3F
6040:3F 3F 3F 3F 00 24 24 24
6048:24 DF DB DB DB 06 36 36
6050:36 36 00 00 00 00 00 00
```

След това да я запишем върху дискетата под името ПРАВОЪГЪЛНИК:

```
BSAVE ПРАВОЪГЪЛНИК, A$6000, L$53
```

Докато таблицата на формите'е в паметта, ще начертаем първата форма, която се съдържа в нея, чрез командите:

```
POKE 232,0: POKE 233,96
HCOLOR = 3: ROT = 0: SCALE = 1: HGR
DRAW 1 AT 100,100
```

В резултат на екрана действително се появява правоъгълник. Може би се питате какво представлява втората форма от таблицата. Тя е *хоризонталното предварително изместване* на правоъгълника, който вече начертахме. За да обясним това сложно на пръв поглед понятие, представете си, че сме изместили правоъгълника с една колона надясно. В този случай основната част от фигурата остава непроменена. Само една допълнителна вертикална линия се добавя отдясно и една се изтрива отляво.За нашия правоъгълник именно тези две линии са предварителното хоризонтално изместване. За да го покажем на екрана, нека начертаем втората форма от таблицата на 10 позиции под първата:

```
DRAW 2 AT 101,110
```

В резултат се изчертава една вертикална линия под колоната, намираща се непосредствено отдясно на правоъгълника, и една линия точно под най-лявата му колона. При така подбраните форми, ако изпълним командата:

```
XDRAW 2 AT 101,100
```

след като правоъгълникът е начертан предварително, той ще се отмести точно една колона надясно.

Как се получава това? За да намерим обяснението, нека си припомним действието на командата XDRAW. Когато се използува за чертане на една форма върху друга, тя сравнява съответните точки от двете форми и образува от тях резултатна фигура. Всяка точка от тази фигура е активна (светеща) в случаите, когато само една от двете препокриващи се точки е била активна. Всъщност при командата XDRAW се изпълнява логическата операция изключващо ИЛИ между битовете, съответстващи на точките от двете фигури. Тази логическа операция действува по следните правила:

точка 1	1	1	0	0
точка 2	1	0	1	0
резултат	0	1	1	0

Когато чертаем предварителното изместване върху правоъгълника с последната команда XDRAW, точките, намиращи се в левите страни на двете фигури, са активни и в резултат от изпълнението на XDRAW се деактивират, т.е. най-левият вертикален ред точки на правоъгълника се изтрива. Предварителното изместване отдясно е с един вертикален ред пред правоъгълника, където всички точки имат съответстващи битове нула. Поради това само точките на предварителното изместване са активни и в резултат от изпълнението на XDRAW всички битове, които съответстват на този ред, се установяват в единица, т.е. този ред се изчертава върху екрана.

За да демонстрираме ефекта от многократното чертане с формата на предварителното изместване, нека изпълним командите:

```
XDRAW 2 AT 101,100
XDRAW 2 AT 102,100
XDRAW 2 AT 103,100
XDRAW 2 AT 104,100
XDRAW 2 AT 105,100
XDRAW 2 AT 105,100
XDRAW 2 AT 104,100
XDRAW 2 AT 103,100
```

В резултат правоъгълникът върху екрана се движи надясно, а след това се връща наляво. Второто изпълнение на XDRAW в точка 105,100 се извършва, за да се елиминира резултатът от първото. Двете еднакви изпълнения се унищожават взаимно, защото с първото правоъгълникът се придвижва надясно, а с второто се връща в предишната си позиция.

```
 1  REM **ПРОГРАМА 6.4**
 2  REM **АНИМАЦИЯ С ПРЕДВАРИТЕЛНО ИЗМЕСТВАНЕ**
10  D$ = CHR$ (13) + CHR$ (4)
20  PRINT D$"BLOAD ПРАВОЪГЪЛНИК"
30  POKE 232,0: POKE 233,96
40  HCOLOR = 3: ROT = 0: SCALE =1
50  HGR : REM 'ЧЕРТАНЕ*
60  DRAW 1 AT 10,100
70  FOR I = 11 TO 270
80  XDRAW 2 AT I,100
90  NEXT I
100 PRINT CHR$ (7): END
```

Програмата 6.4 използува метода на предварителното изместване, за да осъществи хоризонтално движение на правоъгълника. От нейното изпълнение виждаме, че фигурата се движи бързо и плавно. Освен това програмата е проста и кратка. За да я сравните с предишната програма, която чертае и трие цели форми, можете да направите следните промени:

```
60  FOR I = 10 TO 270
70  XDRAW 1 AT I,100
80  XDRAW 1 AT I,100
```

и да я изпълните отново.

Предварителното изместване може да се използува и за осъществяване на вертикално движение. За да се определи предварителното изместване на произволна фигура, се постъпва по следния начин. Първо тя се изчертава с XDRAW. След това се премества в желаната посока и се пречертава отново с XDRAW. Фигурата, която остава на екрана, е желаното предварително изместване. За да демонстрираме този метод, нека да намерим вертикалното изместване на нашия правоъгълник. За целта изпълняваме командите:

```
BLOAD ПРАВОЪГЪЛНИК, A$6000
POKE 232,0: POKE 233,96
XDRAW 1 AT 100,100
XDRAW 1 AT 100,99
```

Фигурата, която се получава на екрана, е *вертикалното предварително изместване* на правоъгълника. За да можете да го използувате, трябва да го включите в таблицата на формите и след това да го чертаете с XDRAW върху правоъгълника. В резултат на това той ще се премества вертикално.

За да покажем използуването на метода на предварителното изместване за осъществяване движение на форми, нека разгледаме програмата 6.5. Тя демонстрира движение на правоъгълника в хоризонтална посока, което се управлява с контролера за игри с номер 0.

```
 1  REM **ПРОГРАМА 6.5**
 2  REM **УПРАВЛЕНИЕ НА ДВИЖЕНИЕТО С КОНТРОЛЕР ЗА
    ИГРИ**
10  D$ = CHR$ (13) + CHR$ (4)
20  PRINT D$"BLOAD ПРАВОЪГЪЛНИК"
30  POKE 232,0: POKE 233,96
40  HCOLOR = 3: ROT = 0: SCALE = 1
50  HGR : REM ПЪРВО ЧЕРТАНЕ
60  DRAW 1 AT 20,100
70  X = 20: SX = 20
80  SP = 1
90  REM *ПРОВЕРКА НА КОНТРОЛЕРА*
100 IF PDL (0) < 90 THEN X = X - 1: NP = 0: GOTO 120
110  IF  PDL (0) > 150 THEN X = X + 1: NP = 1
```

```
120  IF X < 20 OR X > 275 THEN X = SX
130  IF X = SX GOTO 100
140  IF NP < > SP THEN XDRAW 2 AT SX,100
150  XDRAW 2 AT X,100: SX = X: SP = NP
160  GOTO 100
```

В частта си до ред 50 програмата е идентична с програма 6.4. С редове 100 и 110 се проверява стойността на PDL(0) и се увеличава или намалява с 1 координатата X, ако е необходимо. С ред 120 се проверява дали правоъгълникът се намира в границите на екрана. Командата XDRAW от ред 150 чертае предварителното изместване при движение. Тя се прескача с ред 130, ако няма движение. В ред 140 се проверява дали старата посока на движение SP се различава от новата NP. Ако те съвпадат, командата XDRAW се изпълнява повторно със същите координати. При това се елиминира ефектът от предишното изпълнение и се подготвя изображението за движение в обратната посока. Така движението на правоъгълника се подчинява на управлението на контролера за игри с номер 0.

6.3. АНИМАЦИЯ ЧРЕЗ ПРЕМЕСТВАНЕ НА ДАННИ

Таблиците на формите, с които разполага версията на БЕЙСИК за Правец-82, дават добри възможности за изобразяване на движещи се фигури, но за някои игри се оказват недостатъчно бързи. В тези случаи се прибягва до анимация чрез преместване на данни в паметта. Ако трябва да сме по-точни, необходимо е да отбележим, че анимацията с преместване на данни обикновено се реализира от програми, написани на машинен език. Примерите, които ще дадем, са написани на БЕЙСИК, но основната ни цел е да изложим идеите, използувани при този метод. Веднъж усвоени, те могат да бъдат реализирани и на машинен език.

Важно е да се разбере разликата между чертането на фигури чрез преместване на данни в паметта и обикновеното използуване на форми. Както разбрахме, формата се определя от фиксирана последователност от инструкции (вектори). Когато тя се чертае, всяка инструкция се обработва отделно. Поради това, ако формата се състои от много инструкции, векторният метод на построяването ѝ се оказва бавен.

Фигурата, която се чертае чрез преместване на данни, може да се разглежда като форма, която е създадена и начертана предварително някъде в паметта. Чертането чрез преместване на данни се свежда до прехвърляне на информацията за изображението в областта от паметта, която управлява непосредствено екрана. По този начин чертането на форма чрез предест-

ване на данни става по-бързо и се избягва необходимостта да се интерпретират винаги векторите, които я определят.

Програмата 6.6 демонстрира прост пример за анимация чрез преместване на данни в паметта. Ако я въведете в компютъра и я изпълните, ще видите въртяща се стрелка в горния ляв ъгъл на екрана. Този ефект се получава в резултат от многократното повторение на осем различни изображения. Комбинациите от точки (точковите модели), които ги съставят, заедно със съответстващите им двоични и десетични стойности са дадени в табл. 6.1.

Таблица 6.1

А

Точков модел	Стойност	
	Двоична	Десетична
– – – × – – – –	00001000	8
– – – × – – – –	00001000	8
– – – × – – – –	00001000	8
– – – × – – – –	00001000	8
– – – – – – – –	00000000	0
– – – – – – – –	00000000	0
– – – – – – – –	00000000	0

Б

Точков модел	Стойност	
	Двоична	Десетична
– – – – – – – ×	01000000	64
– – – – – – × –	00100000	32
– – – – – × – –	00010000	16
– – – – × – – –	00001000	8
– – – – – – – –	00000000	0
– – – – – – – –	00000000	0
– – – – – – – –	00000000	0

В

Точков модел	Стойност	
	Двоична	Десетична
– – – – – – – –	00000000	0
– – – – – – – –	00000000	0
– – – – – – – –	00000000	0
– – – × × × ×	01111000	120
– – – – – – – –	00000000	0
– – – – – – – –	00000000	0
– – – – – – – –	00000000	0

Г

Точков модел	Стойност	
	Двоична	Десетична
– – – – – – – –	00000000	0
– – – – – – – –	00000000	0
– – – – – – – –	00000000	0
– – – × – – –	00001000	8
– – – – × – –	00010000	16
– – – – – × –	00100000	32
– – – – – – ×	01000000	64

Д

Точков модел	Стойност	
	Двоична	Десетична
– – – – – – – –	00000000	0
– – – – – – – –	00000000	0
– – – – – – – –	00000000	0
– – – × – – –	00001000	8
– – – × – – –	00001000	8
– – – × – – –	00001000	8
– – – × – – –	00001000	8

Е

Точков модел	Стойност	
	Двоична	Десетична
– – – – – – – –	00000000	0
– – – – – – – –	00000000	0
– – – – – – – –	00000000	0
– – – \ – – –	00001000	8
– – \ – – – –	00000100	4
– \ – – – – –	00000010	2
\ – – – – – –	00000001	1

Таблица 6.1 продължение

Ж			З		
Точков модел	Стойност		Точков модел	Стойност	
	Двоична	Десетична		Двоична	Десетична
– – – – – – –	00000000	0	× – – – – – –	00000001	1
– – – – – – –	00000000	0	– × – – – – –	00000010	2
– – – – – – –	00000000	0	– – × – – – –	00000100	4
× × × × – – –	00001111	15	– – – × – – –	00001000	8
– – – – – – –	00000000	0	– – – – – – –	00000000	0
– – – – – – –	00000000	0	– – – – – – –	00000000	0
– – – – – – –	00000000	0	– – – – – – –	00000000	0

```
 1  REM **ПРОГРАМА 6.6**
 2  REM **ВЪРТЯЩА СЕ СТРЕЛКА**
10  REM *ЗАРЕЖДАНЕ НА ДАННИТЕ ЗА ИЗОБРАЖЕНИЕТО*
20  FOR I = 1 TO 8
30  FOR J = 1 TO 7
40  READ V%(I,J)
50  NEXT J,I
60  Y%(1) = 8321:Y%(2) = 9345:Y%(3)
      = 10369:Y%(4) = 11393:Y%(5)
      = 12417:Y%(6) = 13441:Y%(7)
      = 14465: REM АДРЕСИ
80  HGR
90  REM *АНИМАЦИЯ*
100  FOR I = 1 TO 8
110  POKE Y%(4),V%(I,4)
120  POKE Y%(3),V%(I,3)
130  POKE Y%(2),V%(I,2)
140  POKE Y%(1),V%(I,1)
150  POKE Y%(5),V%(I,5)
160  POKE Y%(6),V%(I,6)
170  POKE Y%(7),V%(I,7)
180  NEXT
190  GOTO 100
200  REM *ДАННИ ЗА ИЗОБРАЖЕНИЕТО*
210  DATA 8,8,8,8,0,0,0,64,32,16,8,0,0,0
220  DATA 0,0,0,120,0,0,0,0,0,0,8,16,32,64
230  DATA 0,0,0,8,8,8,8,0,0,0,8,4,2,1
240  DATA 0,0,0,15,0,0,0,1,2,4,8,0,0,0
```

Десетичните стойности са записани в програмата чрез блокове от данни, съдържащи се в редове от 210 до 240. Тези стойности се зареждат в масива V%(8,7) с командата READ (ред 40). Масивът е с размерност 8 X 7, тъй като в него се записва информация за осем положения на стрелката, всяко от които се изобразява със седем байта. В масива Y%(7) се записват адреси-

те на байтовете, управляващи редовете на правоъгълничето, върху което се извежда изображението. Действителната анимация се осъществява в редове 100 – 190. Байтовете, управляващи редовете на правоъгълничето, не се зареждат по възходящ ред на номерата им. Първо се зареждат байтовете, управляващи четвърти, трети, втори и първи ред. След това се зареждат байтовете, управляващи пети, шести и седми ред. Това е направено, за да се постигне по-добър ефект на анимация. Вие можете да промените тази последователност и да видите разликата, която ще се получи.

6.4. ВЕРТИКАЛНО ДВИЖЕНИЕ ЧРЕЗ ПРЕМЕСТВАНЕ НА ДАННИ

Въртящата се стрелка, реализирана от програма 6.6, е пример за статична анимация. При построяването на игри е необходимо повечето фигури да се движат в хоризонтална или вертикална посока.

```
 1  REM **ПРОГРАМА 6.7**
 2  REM **ПАДАЩА ВЪРТЯЩА СЕ СТРЕЛКА**
10  Y1% = 1:Y2% = 2:Y3% = 3:Y4% =4:
    Y5% = 5:Y6% = 6:Y7% = 7
15  REM *ЗАРЕЖДАНЕ НА ДАННИТЕ ЗА ИЗОБРАЖЕ-
    НИЕТО*
20  FOR I = 1 TO 8
30  FOR J = 1 TO 7
40  READ V%(I,J)
50  NEXT J,I
60  REM
70  GOSUB 500
80  HGR
90  REM *АНИМАЦИЯ*
100  FOR I = 1 TO 8
110  POKE Y%(Y4%),V%(I,4)
120  POKE Y%(Y3%),V%(I,3)
130  POKE Y%(Y2%),V%(I,2)
140  POKE Y%(Y1%),V%(I,1)
150  POKE Y%(Y5%),V%(I,5)
160  POKE Y%(Y6%),V%(I,6)
170  POKE Y%(Y7%),V%(I,7)
175 Y1% = Y1% + 1:Y2% = Y2% + 1:Y3% = Y3% + 1:
    Y4% = Y4% + 1:Y5% = Y5% + 1:Y6% = Y6% + 1:
    Y7% = Y7% + 1
177  IF Y7% > 192 THEN END
180  NEXT
190  GOTO 100
```

```
200  REM *ДАННИ ЗА ИЗОБРАЖЕНИЕТО*
210  DATA 8,8,8,8,0,0,0,64,32,16,8,0,0,0
220  DATA 0,0,0,120,0,0,0,0,0,0,8,16,32,64
230  DATA 0,0,0,8,8,8,8,0,0,0,8,4,2,1
240  DATA 0,0,0,15,0,0,0,1,2,4,8,0,0,0
500  REM *ИЗЧИСЛЯВАНЕ  НАЧАЛНИ
     АДРЕСИ НА РЕДОВЕ ТОЧКИ*
510  DIM Y%(192)
520  FOR I = 1 TO 185 STEP 8: READ HA%
530  FOR J = 0 TO 7:Y%(I + J) = HA% + J * 1024
540  NEXT J,I
550  DATA 8192,8320,8448,8576,8704,8832,8960,9088
560  DATA 8232,8360,8488,8616,8744,8872,9000,9128
570  DATA 8272,8400,8528,8656,8784,8912,9040,9168
580  RETURN
```

Програмата 6.7, получена след известни промени на програма 6.6, осъществява движение на въртящата се стрелка във вертикална посока. Новото в нея е подпрограмата, започваща от ред 500. Тя изчислява адресите Y%(192), управляващи най-левите точки на всяка от хоризонталните линии на екрана. За целта към началния адрес HA% на реда правоъгълничета, в който се намира съответният ред точки, се добавя относителният адрес на реда точки. Както видяхме от картата на паметта за режим BPC (фиг. 3.1), този адрес е кратен на 1024 и представлява отместването на реда точки спрямо началния адрес на съответния ред правоъгълничета. Операциите в ред 175 се изпълняват след всяко извеждане на стрелката. Координатата Y се увеличава с единица и следващото извеждане се извършва с един ред по-ниско.

Ако изпълним програмата, ще видим придвижване на въртящата се стрелка надолу в лявата част на екрана. Ще забележим и известни остатъци от нея. По-голямата част на всяко изображение се изтрива от начертаното върху него следващо изображение. Поради движението обаче най-горната линия не се препокрива и оставя неизтрити остатъци върху екрана. Това явление се проявява често при осъществяване на анимация чрез преместване на данни в паметта. В случая то може да се избегне чрез командите:

```
105  POKE Y%(Y0%),0
173  Y0% = Y1%
```

С ред 173 в Y0% се запомня адресът на байта, управляващ най-горния ред от предишното изображение. За да се изтрият остатъците от него, в ред 105 този байт се нулира.

6.5. ХОРИЗОНТАЛНО ДВИЖЕНИЕ ЧРЕЗ ПРЕМЕСТВАНЕ НА ДАННИ

Дотук разгледахме реализирането на статична анимация и движение във вертикална посока чрез преместване на данни. Осъществяването на хоризонтално движение е доста по-трудно. Затова ще дадем следните предварителни обяснения.

Нека си поставим задачата да изведем светеща точка в най-горния ред и най-лявата колона от екрана. За да я изпълним, трябва да променим съдържанието на байта с адрес 8192. Стойността, която трябва да въведем в него, се получава по известния вече начин. Тъй като от всеки байт се извеждат само седем бита, в този случай за най-левите седем точки от най-горния ред на екрана имаме комбинацията:

X– – – – – – –, съответстваща на 00000001 или 1.

За да предизвикаме светенето на най-горната лява точка от екрана, трябва да зададем командите:

```
HGR
POKE 8192,1
```

За да „запалим" последователно втората, третата, четвъртата, петата, шестата и седмата точка от същия байт, трябва да изведем следните комбинации от светещи и несветещи точки (точкови модели), дадени с двоичните и десетичните стойности, които трябва да се съдържат в съответния байт.

Комбинация от точки	Двоична стойност	Десетична стойност
X– – – – – – –	00000001	1
–X– – – – – –	00000010	2
– –X– – – – –	00000100	4
– – –X– – – –	00001000	8
– – – –X– – –	00010000	16
– – – – – X– –	00100000	32
– – – – – – –X	01000000	64

Тези стойности могат да се въведат в адрес 8192 чрез командите на БЕЙСИК

```
HGR
POKE 8192,1: POKE 8192,2
POKE 8192,4: POKE 8192,8
POKE 8192,16: POKE 8192,32
POKE 8192,64
```

В резултат от изпълнението им светещата точка се придвижва надясно и след изпълнението на последната команда спира в най-дясната позиция на полето, управлявано от клетка с адрес 8192. За да продължи движението надясно, точката трябва да премине в полето, което се управлява от съдържанието на адрес 8193.

За целта е необходимо да нулираме клетката с адрес 8192 и да въведем същата последователност от стойности на адрес 8193:

```
POKE 8192,0: POKE 8193,1
POKE 8193,2: POKE 8193,4
POKE 8193,8: POKE 8193,16
POKE 8193,32: POKE 8193,64
```

Така можем да продължим до адрес 8231 = 8192 + 39, в резултат на което точката ще се придвижи по цялата широчина на екрана.

Следователно за всяка светеща точка трябва да въведем определена фиксирана стойност в байта, който я управлява. Всеки байт управлява седем точки от екрана, така че тези стойности са общо седем. При преминаване от един байт в друг последният бит на стария байт остава установен в единица. Той трябва да се нулира допълнително.

За да придвижим светещата точка през определено правоъгълниче на даден ред (фиг. 3.1), необходимо е да намерим отместването на адреса на правоъгълничето спрямо началния адрес на реда, на който се намира. След това в този адрес записваме последователно седемте стойности, придвижващи точката отляво надясно.

Има определена зависимост между координатата X на светещата точка, отместването на адреса на правоъгълничето, в което се намира, и стойността, съдържаща се в управляващия байт. За да я установим, нека номерираме седемте позиции на точките в един байт и съответствуващите им седем стойности с числа от 0 до 6 по следния начин:

позиция:	7	6	5	4	3	2	1
стойност:	64	32	16	8	4	2	1
номер:	6	5	4	3	2	1	0

При тази постановка да се опитаме да „запалим" точката от първия ред на екрана с координата X = 94. Разделяме стойността на координатата на 7. Частното и остатъкът са съответно 13 и 3. Числото 13 е отместването на адреса на байта спрямо основния адрес на реда, на който се намира точката. Числото 3 е номерът на позицията на точката в байта. На нея съответствува стойност 8 за съдържанието на байта. По та-

къв начин за адреса получаваме 8205 = 8192 + 13, а за стойността, която трябва да съдържа 8. Ако сме в режим BPC и изпълним командата

```
POKE 8205,8
```

точката с номер 94 на най-горния ред трябва да започне да свети.

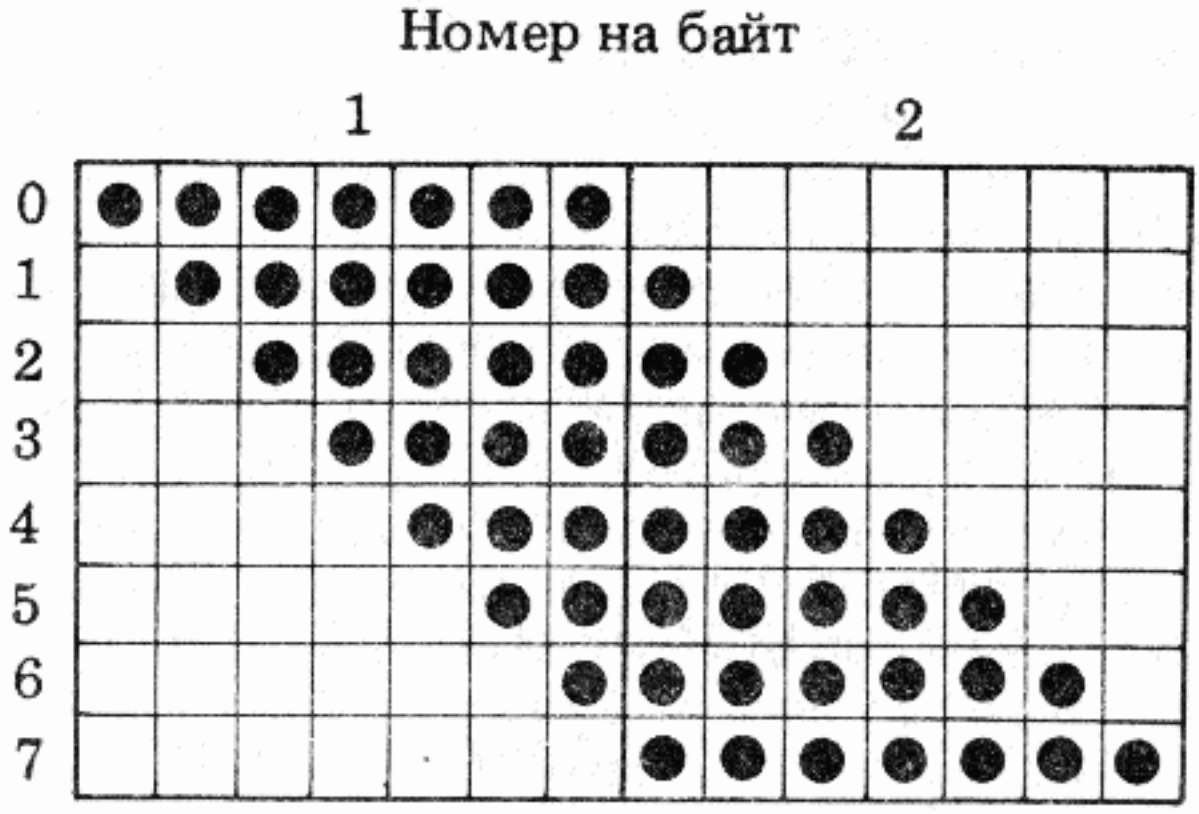
В следващия пример ще „запалим" точка, намираща се на произволно място от екрана. Нека това е точката с координати X = 105 и Y = 64. Редът точки с номер 64 спрямо началото на екрана преминава през най-горната част на правоъгълничетата от осмия ред. Те имат основен адрес 9088. Когато разделим 105 на 7, частното е 15, а остатъкът — 0. Стойността с номер 0 е 1. Затова, за да „запалим" точката с желаните координати, задаваме командата:

```
POKE 9088 + 15,1
```

Сега ще разгледаме няколко примера, при които вместо отделна точка придвижваме хоризонтално отсечка от седем точки. В резултат от изпълнението на програмата 6.8 в горната част на екрана бързо се придвижва къса отсечка, въпреки че е включен забавящ цикъл (ред 50). Движението не е плавно, а на скокове през седем точки. Те се получават, защото на всеки цикъл се изчиства съдържанието на цял байт и в съседния байт се зарежда числото 127, запълващо седемте точки наведнаж (редове 30,40).

```
 1  REM **ПРОГРАМА 6.8**
 2  REM **ХОРИЗОНТАЛНО ДВИЖЕНИЕ НА ОТСЕЧКА**
10  HGR
20  FOR J = 8192 TO 8231
30  POKE J,127
40  POKE J - 1,0
50  FOR I = 1 TO 60: NEXT I
60  NEXT J
```

За да се избегне скокообразното движение, отсечката трябва да се придвижва точка по точка.Нека си представим как отсечката от седем точки ще премине през едно правоъгълниче на екрана. Отначало в лявата страна на правоъгълничето ще се появи само една точка от отсечката. След това ще видим две, три и т.н. до седем точки. В този момент отсечката изцяло ще се помести в правоъгълничето. Оттук нататък тя ще започне да го напуска, като постепенно освобождава точките от лявата му страна. Едновременно с това ще започне да навлиза по същия начин в съседното правоъгълниче отдясно. Тази последователност е показана на фиг. 6.2. Комбинациите от точки (точковите модели), които я реализират, и съответстващите им десетични стойности са дадени в табл. 6.2.



фиг. 6.2

Таблица 6.2

БАЙТ 1		БАЙТ 2	
Десетична стойност	Точков модел	Точков модел	Десетична стойност
127	× × × × × × ×	– – – – – – –	0
126	– × × × × × ×	× – – – – – –	1
124	– – × × × × ×	× × – – – – –	3
120	– – – × × × ×	× × × – – – –	7
112	– – – – × × ×	× × × × – – –	15
96	– – – – – × ×	× × × × × – –	31
64	– – – – – – ×	× × × × × × –	63
0	– – – – – – –	× × × × × × ×	127

Дотук разгледахме двубайтова анимация. Въпреки че отсечката се помества в един байт, за шест от седемте ѝ положения са необходими два байта. Тази особеност на хоризонталната анимация чрез преместване на данни трябва да се има предвид. Всяка фигура, за да се придвижи през седем точки, се представя като седем отделни изображения и изисква един байт повече, отколкото заема неподвижното ѝ изображение. Фигура, заемаща два байта, при хоризонтална анимация изисква три байта и т.н. Единствено фигурата, състояща се от една точка, се събира в един байт.

Ръчно можем да осъществим движението на отсечката чрез следните команди на БЕЙСИК.

```
HGR
POKE 8192,127: POKE 8193,0
POKE 8192,126: POKE 8193,1
POKE 8192,124: POKE 8193,3
POKE 8192,120: POKE 8193,7
```

```
POKE  8192,112:  POKE  8193,15
POKE  8192,96:   POKE  8193,31
POKE  8192,64:   POKE  8193,63
POKE  8192,0:    POKE  8193,127
```

Същото можем да постигнем и с програмата 6.9. Тъй като при нея данните се преместват от компютъра, не представлява трудност да придвижим отсечката по цялата широчина на екрана.

```
 1  REM **ПРОГРАМА 6.9**
 2  REM **ПЛАВНО ДВИЖЕНИЕ  НА ОТСЕЧКА**
20  DIM A%(41)
30  REM *ЗАРЕЖДАНЕ НА ИЗОБРАЖЕНИЕТО*
40  FOR I = 0 TO 6
50  READ H%(I),K%(I)
60  NEXT I
80  HGR
90  REM *ЗАРЕЖДАНЕ НА АДРЕСИ*
100  J = 0
110  FOR I = 8616 TO 8655
120  A%(J) = I:J = J + 1
130  NEXT
140  REM *АНИМАЦИЯ*
150  FOR X = 1 TO 280
160  CH% = INT (X / 7)
170  O% = X - (7 * CH%)
180  C% = CH% + 1
190  POKE A%(CH%),K%(O%):POKE A%(C%),H%(O%)
200  NEXT X: END
220  REM  *ДАННИ ЗА ИЗОБРАЖЕНИЕТО*
230  DATA 1,126, 3, 124,7,120, 15, 11, 2, 31, 96, 63, 64, 127, 0
```

Стойностите за изобразяване на различните положения на отсечката се зареждат в масивите H%(6) и K%(6) съответно за началото и края на отсечката. Това се извършва в редове 40 – 60, като всяка двойка стойности се чете от блока данни, който се съдържа в ред 230. Масивът A%(41) съдържа адресите на 40-те правоъгълничета от ред 12 на екрана. Той се зарежда в редове 100 – 130 от програмата. Командите от редове 150 – 200 изчертават отсечката за всяка координата X. С командите вътре в цикъла се определят адресите на двойката байтове, която се обработва в момента, и стойностите,които трябва да се заредят в тях. За целта се изчисляват частното CH% и остатъкът O% от делението на координатата X на 7. В променливата CH% се определя адресът на байта, съдържащ крайната част на отсечката, а в C% = CH% + 1 се определя адресът на следващия байт, съдържащ началото ѝ. Остатъкът O% съдържа

номера на стойностите, които трябва да се заредят в двата байта. Този номер се използува като индекс на масивите H%(6) и K%(6), в които са записани самите стойности. Действителното зареждане на байтовете (пренасянето на данни от една област на паметта в друга) се извършва в ред 190.

При осъществяване на движение на фигура чрез преместване на данни е необходимо да се определят предварително седемте положения, заемани от нея по време на движението ѝ през едно правоъгълниче от екрана. След това, както направихме за отсечката, трябва да се определят двоичните и десетичните стойности за всяко от тези положения. Така получените данни се съхраняват в таблица, за да се използуват впоследствие при осъществяване на анимацията.

От всичко това се разбира, че хоризонталното движение, осъществено чрез преместване на данни, изисква значителна предварителна подготовка,особено при по-сложни фигури. За сметка на това, анимацията, постигната по този начин, е изключително плавна, макар и сравнително бавна.

Забавянето се получава от изчисленията, извършвани при всяко преместване на отсечката с една точка. В нашия пример то не се отразява на плавността на движението, защото фигурата е сравнително проста. При по-сложни фигури необходимостта от дълги изчисления може да увеличи времето, през което фигурата се задържа на екрана, и с това да стане забележим моментът на преместването ѝ (изтриването и преначертаването ѝ).

За да избегнем този недостатък, можем да използуваме метода на предварителното изчисление, който разгледахме при осъществяване на анимация чрез командата HPLOT. За случая на хоризонтална анимация чрез преместване на данни пътят на обекта се изчислява преди започване на неговото движение. Получените координати се съхраняват в масив и се използуват непосредствено при осъществяване на анимацията. Очевидно много по-бързо е да се прочете едно число от масив (особено ако се използува програма на машинен език), отколкото да се изчислява в момента на чертане.

```
 1  REM **ПРОГРАМА 6.10**
 2  REM **УСКОРЕНО ДВИЖЕНИЕ НА ОТСЕЧКА**
20  DIM A%(41)
30  REM *ЗАРЕЖДАНЕ НА ИЗОБРАЖЕНИЕТО*
40  FOR I = 0 TO 6
50  READ H%(I),K%(I)
60  NEXT I
80  J = 0: REM ЗАРЕЖДАНЕ НА АДРЕСИ
90  FOR I = 8616 TO 8655
100  A%(J) = I:J = J + 1
```

```
110  NEXT
120  REM *ПРЕДВАРИТЕЛНО ИЗЧИСЛЯВАНЕ*
130  DIM CH%(280),O%(280)
140  FOR X = 1 TO 280
150  CH%(X) = X / 7:O%(X) = X - CH%(X) * 7
160  NEXT X
180  HGR
190  REM *АНИМАЦИЯ*
200  FOR X = 1 TO 280
210  POKE A%(CH%(X)),K%(O%(X)):
     POKE A%(CH%(X) + 1),H%(O%(X))
220  NEXT X: END
240  REM  *ДАННИ ЗА ИЗОБРАЖЕНИЕТО*
250  DATA 1,126,3,124,7,120,15,112,31,96,63,64,127,0
```

С програмата 6.10 се реализира движение на същата отсечка, като се използува методът на предварителното изчисление. В сравнение с програма 6.9 нови са редовете 130, 140, 150 и 210. Масивите CH% и O% съдържат частните и остатъците за всяка от 280-те координати X на реда, получени в резултат от делението им на 7 (редове 140 – 160). Масивът A%(41) отново съдържа адресите на 40-те правоъгълничета от ред 12 на екрана. С ред 210 се извършва действителното зареждане на информацията за изображението в предварително изчислените адреси.

С тази програма отсечката се придвижва по цялата широчина на екрана за седем секунди. Предишната програма върши това за девет секунди. При по-сложни фигури разликата във времената е по-съществена.

Анимацията чрез преместване на данни може да се реализира в комбинация с метода на предварителното изместване, който разгледахме при анимацията чрез таблици на форми. Първоначално се изчертава цялата фигура. След това се използуват предварителните измествания, за да се осъществи движението ѝ. За разлика от предишната програма вместо седем изображения на фигурата се използуват седем предварителни измествания. Подходящото предварително изместване се избира в зависимост от остатъка, получен от делението на координатата X на 7.

Функцията на командата XDRAW в този случай се осъществява чрез логическата операция изключващо ИЛИ. Тя сравнява двойка битове и дава резултат 1, ако те са различни, и 0, ако са еднакви. Затова, ако извършим операция изключващо ИЛИ между един байт, съдържащ само единици (11111111), и друг байт с произволно съдържание, в резултат получаваме инвертираното съдържание на втория байт.

Изпълнението на изключващо ИЛИ с командите на БЕЙСИК

изисква съставянето на специална подпрограма. Машинният език разполага с инструкция,която я осъществява непосредствено.

6.6. АНИМАЦИЯ ЧРЕЗ ЧАСТИЧНИ ПРОМЕНИ

Основната идея на този метод личи от името му. Вместо да се чертае винаги цялата фигура при извеждане на отделните „кадри" на изображението, се чертаят само онези части от фигурата, които са се променили в сравнение с предишното ѝ положение. Често този метод се използува при извеждането на числовите резултати на играта. Както знаем, цифрите се изменят (следват една след друга) по предварително определен начин. Това се използува, за да се намали броят на байтовете, извеждани при изобразяването на всяка следваща цифра.

За да демонстрираме този метод, нека изпълним програмата 6.11. В резултат ще видим, че на екрана се редуват цифрите 0, 3, 8 и 9. Това се осъществява, като се записват подходящи стойности в адресите от паметта, управляващи изображението на правоъгълничето, разположено точно в средата на екрана (X = 20 и Y = 12).

```
 1  REM **ПРОГРАМА 6.11**
 2  REM **ИЗОБРАЗЯВАНЕ НА ЦИФРИ 0, 3, 8 И 9**
 5  HGR
10  REM *ИЗОБРАЗЯВАНЕ НА НУЛА*
20  POKE 8635,60: POKE 9659,66:
    POKE 10683,66: POKE 11707,66
30  POKE 12731,66: POKE 13755,66:
    POKE 14779,60: POKE 15803,0
40  FOR I = 1 TO 300: NEXT
50  REM *ИЗОБРАЗЯВАНЕ НА ТРИ*
60  POKE 8635,60: POKE 9659,64:
    POKE 10683,64: POKE 11707,60
70  POKE 12731,64: POKE 13755,64:
    POKE 14779,60: POKE 15803,0
80  FOR I = 1 TO 300: NEXT
90  REM *ИЗОБРАЗЯВАНЕ НА ОСЕМ*
100  POKE 8635,60: POKE 9659,66:
     POKE 10683,66: POKE 11707,60
110  POKE 12731,66: POKE 13755,66:
     POKE 14779,60: POKE 15803,0
120  FOR I = 1 TO 300: NEXT
130  REM *ИЗОБРАЗЯВАНЕ НА ДЕВЕТ*
140  POKE 8635,60: POKE 9659,66:
     POKE 10683,66: POKE 11707,60
150  POKE 12731,64: POKE 13755,64:
```

```
      POKE 14779,60: POKE 15803,0
160   FOR I = 1 TO 300: NEXT
170   GOTO 10
```

Комбинациите от светещи точки, изобразяващи цифрите 3 и 9 в рамките на едно правоъгълниче, са показани в табл. 6.3.

Таблица 6.3

Точков модел	Стойност		Точков модел	Стойност	
	Двоична	Десетична		Десетична	Двоична
– – × × × × –	00111100	60	– – × × × × –	00111100	60
– – – – – – ×	01000000	64	– × – – – – ×	01000010	66
– – – – – – ×	01000000	64	– × – – – – ×	01000010	66
– – × × × × –	00111100	60	– – × × × × –	00111100	60
– – – – – – ×	01000000	64	– – – – – – ×	01000000	64
– – – – – – ×	01000000	64	– – – – – – ×	01000000	64
– – × × × × –	00111100	60	– – × × × × –	00111100	60
– – – – – – –	00000000	0	– – – – – – –	00000000	0

Дадени са и съответните двоични и десетични стойности, които ги установяват. Ако сравним поредицата от стойности, съответствуващи на цифрите 3 и 9, ще видим, че само две от седемте стойности се различават. Подобно е положението, ако сравним стойностите, съответствуващи на 8 и 9. Разликата между 9 и 0 е в три стойности и т.н. Тогава възниква въпросът: тъй като 9 обикновено следва 8, а 0 следва 9, има ли смисъл да извеждаме целите фигури, изобразяващи тези цифри? В отговора на този въпрос се съдържа идеята на анимацията с частични промени. За да я използуваме, в програмата 6.11 е необходимо да променим следните редове:

```
55   REM  ОТ 0 КЪМ 3
60   POKE 9659,64: POKE 10683,64: POKE 11707,60
70   POKE 12731,64: POKE 13755,64
95   REM ОТ 3 КЪМ 8
100  POKE 9659,66: POKE 10683,66
110  POKE 12731,66: POKE 13755,66
140  REM ОТ 8 КЪМ 9
150  POKE 12731,64: POKE 13755,64
```

Вижда се, че най-много промени се извършват при преминаването от 0 към 3, по-малко от 3 към 8 и най малко от 8 към 9.

Повече време и усилия се икономисват при използуване на метода на частичните промени за изобразяване на по-сложни фигури.

ГЛАВА 7. ЗВУК И МУЗИКА

7.1. ПРОИЗВЕЖДАНЕ НА ЗВУК

Съществуват два основни начина за произвеждане на звук в микрокомпютъра Правец-82. Това са програмни обръщения към клетки с адреси −16336 и −16352, които съдържат програмни ключове с две състояния.

Клетката с адрес −16336 управлява електронни схеми, въздействуващи върху мембраната на високоговорителя, вграден в микрокомпютъра. Клетката с адрес −16352 е свързана със схемите, от които се получават електрически сигнали за запис на информация върху външно запомнящо устройство на магнитна лента (касетофон).

Еднократното обръщение към адрес −16336 превключва съответния програмен ключ от едно състояние в друго. Както е показано на фиг.7.1, това променя посоката на електрическия ток, протичащ през бобината на високоговорителя. Тъй като тя се намира под действието на силно магнитно поле, смяната на посоката на тока предизвиква смяна на посоката на силите, с които то ѝ въздействува. В резултат бобината се премества бързо от едно крайно положение в друго. Заедно с нея се премества и мембраната на високоговорителя (те са свързани механически), която при рязкото си движение въздейства на околния въздух. Получават се звукови вълни, които човешкото ухо възприема като слабо пукане.

Ако в програма извършим поредица обръщения към адрес −16336, мембраната на високоговорителя трепти с честотата на тези обръщения. В резултат се образуват звукови вълни със същата честота.

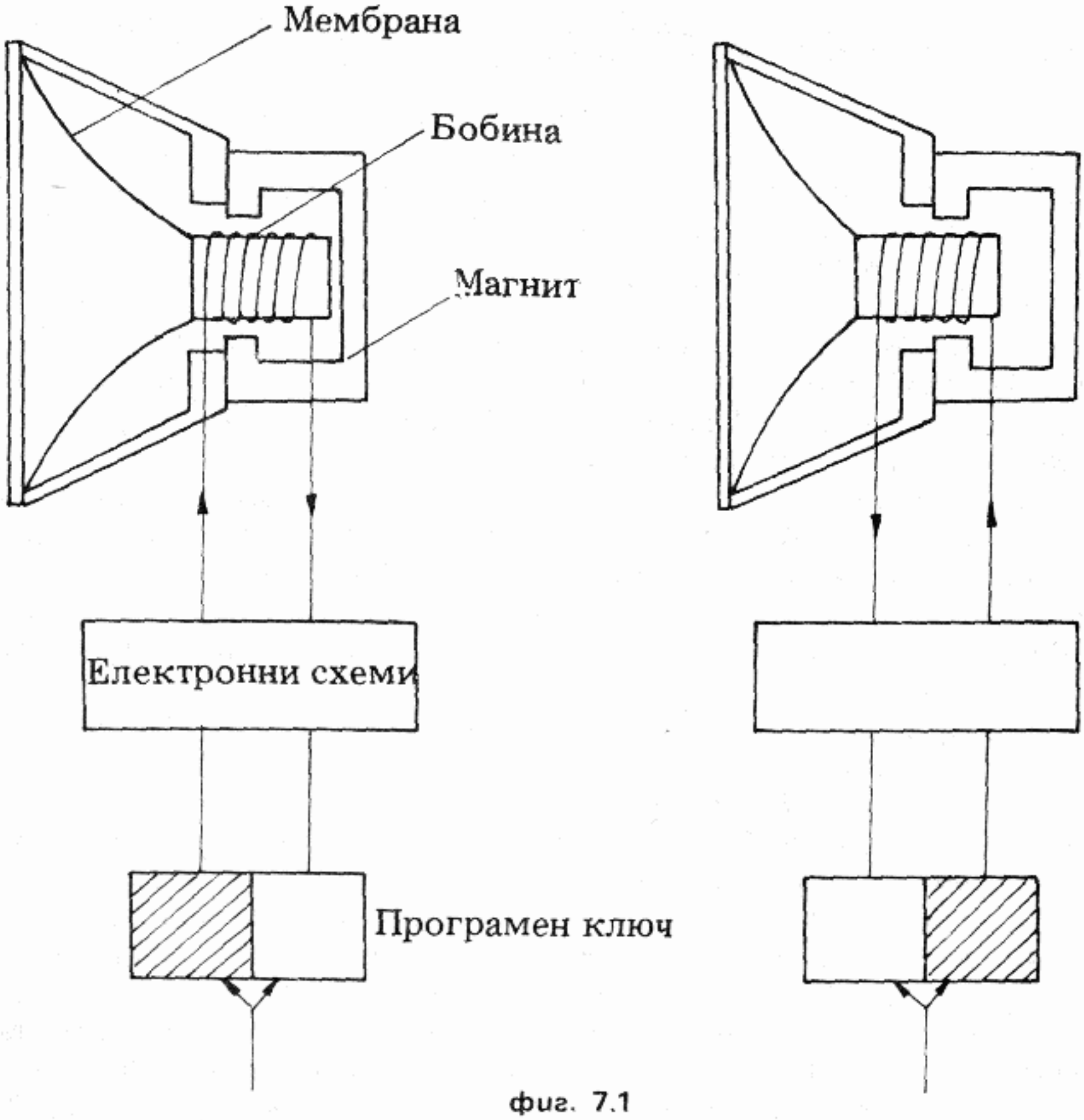
От физиката е известно, че височината на звука се определя от честотата на звуковите трептения. Следователно, ако променяме честотата на обръщенията към адрес −16336, ще променяме и височината на произвеждания звук. Програмно това се реализира, като се променя времето за изпълнение на частта от програмата между командите, извършващи две последователни обръщения. Например с командите на БЕЙСИК

```
10 FOR I = 1 TO 100 : X = PEEK (−16336) : NEXT
```

се получава звук, наподобяващ музикален тон. С командите от следващите два реда

```
10 H = − 16336
20 FOR I = 1 TO 100 : X = PEEK(H) : NEXT
```

се произвежда звук с по-голяма височина. Това се получава, за-



фиг. 7.1

щото в първия случай интерпретаторът на БЕЙСИК изразходва на всяко обръщение допълнително време за преобразуване на числото −16336 в число с плаваща запетая. Във втория случай това преобразуване се извършва еднократно само в ред 10. През време на цикъла (ред 20) числото с плаваща запетая се взема непосредствено от паметта, като се използува променливата H. Това намалява периода на обръщение към адрес −16336 и съответно увеличава честотата.

Този пример ни подсказва, че можем да регулираме честотата на обръщение в програмата, като вмъкваме закъснения с различна продължителност. Например програмата 7.1 съдържа един главен цикъл (ред 20), в който променливата I се променя от 1 до 40. При всяко преминаване през цикъла се извършва обръщение към адрес −16336. Времето между тези обръщения се променя (нараства), защото в главния цикъл е вмъкнат вътрешен цикъл (ред 30), внасящ променящото се закъснение. При всяко преминаване през главния цикъл продължителността на

вътрешния цикъл нараства. По този начин се получава звук с постепенно намаляваща честота.

```
1  REM * * ПРОГРАМА 7.1 * *
2  REM * * ПОНИЖАВАЩ ЗВУК * *
10 H =  -16336
20  FOR I = 1 TO 40: X =  PEEK (H)
30  FOR J = 1 TO I: NEXT
40  NEXT I
50  END
```

В програмата 7.2 е осъществен обратният ефект — повишаване на честотата на звука. Това се постига също чрез вграждане на цикли. Разликата е в това, че вътрешният цикъл е с намаляваща продължителност. За да се постигне малко по-силен звук, командата X = PEEK (H) от предишната програма е заменена с командата X = PEEK (H) + PEEK (H).

```
1  REM * * ПРОГРАМА 7.2 * *
2  REM * * ПОВИШАВАЩ ЗВУК * *
10 H =  -16336
20  FOR I = 35 TO 1 STEP  - 1:X =
    PEEK (H) +  PEEK (H)
30  FOR J = 1 TO I: NEXT
40  NEXT I
50 END
```

С програмата 7.3 се произвежда звук със случайно променяща се честота. Това е постигнато, като в главния цикъл, извършващ обръщение към адрес −16336, е вмъкнат вътрешен цикъл, осъществяващ закъснения със случайна продължителност. Използувана е стандартната функция на БЕЙСИК RND (1). При изпълнението ѝ се получават случайни числа в границите между 0 и 0,999999999.

```
1  REM  * * ПРОГРАМА 7.3 * *
2  REM  * * ШУМ * *
10 H =  -16336
20  FOR I = 1 TO 500
30 X = PEEK (H): FOR J = 1 TO RND
   (1) * 30: NEXT
40  NEXT I
50  END
```

7.2. УСИЛВАНЕ НА ЗВУКА

Ако искаме да усилим звука, получен от микрокомпютъра, необходимо е вместо към адрес −16336 да извършваме обръщения към адрес −16352. Освен това трябва да свържем усилвателната уредба към изхода на микрокомпютъра, от който се извежда сигналът за запис на информация върху касетофон.

Еднократното обръщение към адрес −16352 предизвиква смяна на нивото на електрическия сигнал на изхода за касетофон (0 − 25 mV). Ако извършим многократни обръщения с определена честота, нивото на сигнала на този изход се променя със същата честота. Затова, ако усилим получения сигнал и го подадем към високоговорителя на външна звукова уредба, ще произведем звук с честота, съвпадаща с честотата на обръщенията към адрес −16352. Тъй като звуковата уредба има значително по-добри честотни параметри от високоговорителя на микрокомпютъра, може да се очаква, че получената по този начин музика ще бъде по-качествена.

Всички примери, дадени в тази глава, извършват обръщения към адрес −16336. Същите разглеждания могат да се отнесат към адрес −16352.

7.3. СИГНАЛ „ЗВЪНЕЦ"

Микрокомпютърът Правец-82 произвежда звук, наподобяващ клаксон на автомобил. Този сигнал се получава, като се натиснат едновременно клавишите МК и Г/G. Тогава се включва специална подпрограма от системния МОНИТОР, извършваща последователни обръщения към адрес −16336 с честота 1 kHz. Тази подпрограма се нарича ЗВЪНЕЦ. Със същото име се означава и кодът, изпращан от клавиатурата към микрокомпютъра при едовременното натискане на клавишите МК и Г/G. Въпреки че произвежданият звук не наподобява звука на звънец, той носи това име по исторически причини. То е останало от времето, когато терминалите от телетайпен тип действително са задвижвали малък звънец при получаване на кода „звънец".

Този сигнал може да се използува при съставяне на програми за игри. С него се сигнализира например за груби грешки или верни попадения, като се използува заедно с командата PRINT. Тъй като едновременното натискане на МК и Г/G не се отбелязва със знак на екрана или върху печатащото устройство, употребата на сигнала „звънец" по този начин не оставя видими следи в програмите. Например, докато не изпълним командата:

```
PRINT "ВИЕ УЛУЧИХТЕ"
```

не можем да преценим дали тя включва кода „звънец". Това не пречи на правилната работа на програмата, но затруднява разчитането ѝ. Затова сигналът „звънец" се задава по други на-

чини. Функцията CHR$(7) генерира същия код и се отбелязва върху екрана, т.е. можем да запишем

```
PRINT "ВИЕ УЛУЧИХТЕ"; CHR$ (7)
```

или

```
PRINT CHR$ (7);"ВИЕ";CHR$ (7);"УЛУЧИХТЕ";CHR$ (7)
```

Изписването на цялата последователност от символи "CHR$(7)" е отегчително. По-удобно е в началото на програмата да се дефинира символна променлива, на която да се присвои стойността на кода „звънец". Променливата се избира с кратко име, напомнящо предназначението ѝ, и се използува в останалата част от програмата вместо функцията CHR$(7). Този подход е приложен в следния пример:

```
10  Z$ = CHR$ (7)
20  PRINT Z$;"ВИЕ";Z$;"УЛУЧИХТЕ";Z$
```

Ако изпълните тази програма, ще забележите, че думите и звукът се появяват почти едновременно. Командата PRINT се изпълнява толкова бързо, че не можем да разберем дали думите следват звука или обратно. Ако зададете кода „звънец" няколко пъти, ще забележите, че отделните звуци се сливат и с командата PRINT Z$;Z$;Z$;Z$; се получава непрекъснат удължен звук.

Командата SPEED управлява скоростта на извеждане на знаците от командата PRINT. Например SPEED = 255 определя най-бързо извеждане, а SPEED = 0 – най-бавно. При липса на SPEED се подразбира скорост 255. Програмата 7.4 показва интересен начин за регулиране на закъсненията между отделните звуци „звънец" чрез SPEED. Изпълнете я първоначално, като установите най-бавна скорост на извеждане чрез SPEED = 0. След това повторете изпълнението, като променяте скоростта в ред 20 на 50, 100, 150 и т.н. Ред 40 е включен, за да се възстанови нормалната скорост на извеждане 255.

```
1  REM ** ПРОГРАМА 7.4**
2  REM ** УПРАВЛЕНИЕ НА СИГНАЛА "ЗВЪНЕЦ" **
10 Z$ =  CHR$ (7)
20  SPEED = 0
30  PRINT Z$;Z$;Z$
40  SPEED = 255
```

Сигналът „звънец" може да се използува при съставяне на прости програми. Например с програмата 7.5 децата могат да се учат да броят. Тя предоставя възможност на детето да избере число в определени граници, примерно от 0 до 10. След това сигналът „звънец" се извежда толкова пъти, колкото е зададеното число. В същото време на екрана се появяват цифри, отброяващи изведените до момента сигнали. По този начин детето може да брои сигналите, като гледа върху екрана числата, съответстващи на броя им.

Лесно можем да променим максималното число, до което се брои, като в ред 20 запишем ново число.

```
 1  REM **ПРОГРАМА 7.5**
 2  REM ** БРОЕНЕ НА СИГНАЛА "ЗВЪНЕЦ"**
10 L = 1
20 M = 10
30  TEXT : HOME
40  PRINT "МОЛЯ ИЗБЕРЕТЕ ЧИСЛО МЕЖДУ"
    ;L;" И ";M;: INPUT ": "; N
50  IF N  < L OR N > M THEN 30
60  PRINT
70  PRINT "БРОЙТЕ СИГНАЛИТЕ": PRINT: PRINT
80  FOR J = 1 TO 1000: NEXT : REM
    ПАУЗА ПРЕДИ ПЪРВИЯ СИГНАЛ
90  FOR J = 1 TO N
100  SPEED = 0
110  PRINT "  ":REM ДВА ИНТЕРВАЛА МЕЖДУ ЦИФРИТЕ
120  SPEED = 255
130  PRINT J;
140  PRINT  CHR$ (7);
150  NEXT
160  PRINT : VTAB 18
170  PRINT "НАТИСНЕТЕ RETURN, ЗА
    ДА ОПИТАТЕ ПАК"
175  PRINT "ЗА ДА ЗАВЪРШИТЕ – ОСВ";
180  GET W$
190  IF W$ = CHR$ (27) THEN END:
     REM ПРОВЕРКА ЗА КРАЙ ЧРЕЗ ОСВ
200  GOTO 30
```

7.4. ПРОИЗВЕЖДАНЕ НА МУЗИКАЛНИ ТОНОВЕ

Въпреки че сигналът „звънец" може да се използува за обмен на информация, той не предоставя разнообразни възможности. В крайна сметка при неговото приложение се повтаря, макар и в различни последователности, един и същ звук. Произвеждането на звук чрез непосредствено обръщение към адрес –16336 от програми на БЕЙСИК също има ограничени възможности. Затруднението идва от това, че те се изпълняват сравнително бавно и не могат да произвеждат добри музикални тонове.

За да се преодолее този недостатък, се използуват програми, написани на машинен език. Те са кратки и се извикват от програми на БЕЙСИК, когато трябва да се произведе музикален тон.

```
1  REM **ПРОГРАМА 7.6**
2  REM **ЗВУК ЧРЕЗ МАШИННА ПОДПРОГРАМА**
10  FOR I = 770 TO 790: READ J: POKE
    I,J: NEXT : REM ЗАРЕЖДАНЕ
20  HOME
30  VTAB 6: INPUT "ВИСОЧИНА:";H
40  PRINT : INPUT "ПРОДЪЛЖИТЕЛНОСТ:";D
50  POKE 768,H: POKE 769,D: CALL
    770: REM ИЗПЪЛНЕНИЕ
60  GOTO 20
90  REM *  МАШИННА ПОДПРОГРАМА *
100 DATA 173,48,192,136,208,5,20
    6,1,3,240,9,202,208,245,174,
    0,3,76,2,3,96
```

Програмата 7.6 е прост пример за използуване на този метод. В ред 100 тя съдържа машинната програма, представена с десетични числа. Те се четат от командата READ и се зареждат в клетки с адреси 770 – 790. Разположението на шестнадесетичните им еквиваленти в тази част от паметта е показано в табл.7.1. Чрез променливите H и D в програмата на БЕЙСИК се задават височината и продължителността на произвеждания звук. Техните стойности се записват на адреси 768 и 769 ($300 и $301), откъдето се използуват при изпълнение на машинната програма.

Необходимо е да отбележим, че има множество варианти за съставяне на машинни програми, произвеждащи качествени звуци. Представената в табл.7.1 програма е една от най-разпространените и включва основните елементи, съдържащи се в останалите. В повечето литературни източници машинните програми са представени като „черни кутии" или тайнствени поредици от числа, без да е показан записът им на асемблерен език. Това не пречи те да се използуват, ако не смятате за необходимо да вникнете в механизма на тяхното действие. За по-любознателните читатели и тези, които познават АСЕМБЛЕР за микропроцесора 6502, ще разгледаме представената машинна програма. Ще дадем зависимостта между стойността, зареждана в адрес 768 ($300), и височината на произвеждания звук.

От табл.7.1 се вижда, че по-голямата част от основния период между две обръщения към адрес –16336 се заема от времето, за което се извършват H изпълнения на цикъла от четири инструкции: BNE $030D, DEX, BNE $0305, DEY. За всяко обръщение към адрес –16336 инструкцията LDX $0300 зарежда съдържанието на променливата H (записано предварително на адрес $300) в регистъра X. Съдържанието на регистъра X се намалява с единица при всяко изпълнение на инструкцията DEX и се про-

Таблица 7.1

Шестна-десетичен адрес	Машинна програма	Асемблерна програма	Описание	Брой маш. такта
0302	AD 30 C0	LDA $C030	Зарежда акумулатора от адрес $C030(−16336). Задвижва мембраната на високоговорителя.	4
0305	88	DEY	Намалява с 1 съдържанието на регистъра Y.	2
0306	D0 05	BNE $030D	Преход към адрес $030D,ако Y не съдържа нула.	3
0308	CE 01 03	DEC $0301	Намалява с 1 съдържанието на адрес $301.	6
030B	F0 09	BEQ $0316	Преход към адрес $316, ако клетката $301 съдържа нула.	3
030D	CA	DEX	Намалява с 1 съдържанието на регистъра X.	2
030E	D0 F5	BNE $0305	Преход към адрес $305, ако X не съдържа нула.	3
0310	AE 00 03	LDX $0300	Зарежда регистъра X от адрес $300.	4
0313	4C 02 03	JMP $0302	Безусловен преход към адрес $302.	3
0316	60	RTS	Връщане към програмата на БЕЙСИК.	6

верява дали е нула от инструкцията BNE $0305. Когато това условие се изпълни, регистърът X се зарежда отново със стойността на H и се извършва обръщение към адрес −16336. Тъй като всяка от инструкциите, DEX и DEY, се изпълнява за два машинни такта, а инструкциите BNE $030D и BNE $0305 за 3, целият цикъл се изпълнява за десет такта. Тогава времето за H изпълнения на цикъла е равно на $t_1 = 10Ht_{МТ}$, където $t_{МТ}$ е продължителността на един машинен такт.

Останалата част от основния период между две обръщения към адрес −16336 се заема от времето за еднократно изпълнение на инструкциите LDX $0300, JMP $0302 и LDA $C030. То е равно на 11 машинни такта или : $t_2 = 11t_{МТ}$.

Машинният такт е времето, за което микрипроцесорът изпълнява един пълен цикъл на информационен обмен с паметта (операция четене или запис). За микропроцесора 6502 той е равен приблизително на 1μs (10^{-6}s).

Тъй като едно звуково трептене се създава за две обръщения към адрес −16336, неговият период, изразен чрез съдържанието

на променливата H, е равен на

$T = 2(t_1 + t_2) = 2(10Ht_{мТ} + 11t_{мТ}) = 2(10H + 11).10^{-6}$.

Връзката между честотата f и периода T на звуковите трептения се определя от израза

$T = 1/f$.

От двете зависимости получаваме

$1/f = 2(10H + 11).10^{-6}$, откъдето намираме

$f = 1\ 000\ 000/2(10H + 11)$.

За да бъде зависимостта по-точна заместваме 1 000 000 с точната стойност на тактовата честота, която за микропроцесора 6502 е 1 023 000 Hz(1.023 MHz). Тогава зависимостта между съдържанието на променливата H и честотата f се дава с израза:

$f = 1\ 023\ 000/2(10H + 11)$.

Зависимостта между продължителността на звука и съдържанието на променливата D може да се определи по аналогичен начин. Преди стартиране на програмата стойността на D се записва на адрес 769 ($301). Всяко изпълнение на инструкция DEC $0301 намалява съдържанието на този адрес с единица, а с инструкция BEQ $0316 се проверява дали е равно на нула. Когато това условие се удовлетвори, изпълнението на програмата се прекратява. Инструкциите DEC $0301 и BEQ $0316 се изпълняват на всяко 256-то изпълнение на инструкция DEY. Това е направено, за да могат да се задават приемливи за практиката продължителности с числа, поместващи се в един байт (0 – 255).

Влиянието на съдържанието на адреси 768 и 769 върху височината и продължителността на звука може да се провери чрез програмата 7.7. Съдържанието на клетки 768 и 769 се променя последователно от 10 до 250 със стъпка 10. При изпълнение на програмата се чува звуковият ефект от тези промени.

```
1  REM**ПРОГРАМА 7.7**
2  REM ** ПРОМЯНА НА ВИСОЧИНАТА И
   ПРОДЪЛЖИТЕЛНОСТТА НА ЗВУКА**
10  FOR I = 770 TO 790: READ J: POKE I,J: NEXT
15 REM * ПРОМЯНА НА ВИСОЧИНАТА*
20  FOR I = 10 TO 250 STEP 10: POKE
    768,I:  POKE  769,64
30  CALL 770: NEXT
35 REM* ПРОМЯНА НА ПРОДЪЛЖИТЕЛНОСТТА*
40  FOR I = 10 TO 250 STEP 10: POKE
     769,I:  POKE  768,64
45  FOR J = 1 TO 200: NEXT
50  CALL 770: NEXT
60  DATA 173,48,192,136,208,5,206
     ,1,3,240,9,202,208,245,174,0
     ,3,76,2,3,96
```

В програмата 7.8 съдържанието на адреси 768 и 769 се променя по случаен закон и при изпълнението ѝ се получава ... „компютърна песен".

```
 1  REM **ПРОГРАМА 7.8**
 2  REM **КОМПЮТЪРНА ПЕСЕН**
10  FOR I = 770 TO 790: READ J: POKE I,J: NEXT
20  FOR I = 1 TO 150: POKE 768, RND
    (3) * 255: POKE 769, RND (3) * 10 + 1
30  CALL 770: NEXT
100  DATA 173,48,192,136,208,5,206,
    1,3,240,9,202,208,245,174,
    0,3,76,2,3,96
```

7.5. МУЗИКАЛЕН СТРОЙ

Както е известно от теория на музиката, *музикален строй (звуков ред)* се наричат абсолютните височини на звуците в дадена музикална система и съотношението помежду им.

Строят се основава на височината на един звук, който се използува за ориентир. Обикновено това е тонът ла от първа октава, за който са приети официално 440 трептения в секунда. На основата на звука-ориентир се определят височините на всички останали звуци в музикалната система. За тази цел в продължение на историческото развитие на музиката са използувани различни методи. Сега в повечето страни от света се използува *дванадесетзвуковата равномерна темперация.*

Дванадесетзвуковият темперирaн строй разделя октавата на дванадесет равни части, наречени полутонове. *Полутонът* е най-малкото разстояние по височина между два съседни звука.

За да се възпроизведат музикалните тонове на дванадесетзвуковия строй с машинната програма от табл. 7.1, необходимо е да се намерят онези стойности на променливата H, които произвеждат възможно най-близките честоти до тези тонове.

За разлика от обикновените музикални инструменти, чиято настройка се извършва по слух, честотите, произвеждани от компютъра, се определят на основата на математически зависимости. При това съществуват известни ограничения. Тъй като числовите стойности, записани на адрес 768, не могат да следват точно честотните отношения на музикалния строй, някои от нотите се възпроизвеждат малко отместени. Това важи особено за горните октави, където разстоянието между височините е по-малко. Получените по този начин звуци не съответствуват точно на височините на музикалните тонове, но поради това, че имат правилно честотно отместване помежду

си, общо звучат музикално. С други думи, микрокомпютърът самостоятелно свири правилно, но не може да свири заедно с оркестър.

Програмата 7.9 съдържа числовите стойности, задаващи височината на 49 ноти. Те покриват четири октави и един полутон от музикалния строй. Записани са в редове 300 – 330, откъдето се зареждат с команда READ (ред 30) в масива H(49). От там се прехвърлят последователно в адрес 768 и се изпълняват.

```
1   REM ** ПРОГРАМА 7.9**
2   REM ** НОТИ**
10  FOR I = 770 TO 790: READ J: POKE I,J: NEXT
20  READ N: DIM H(I.)
30  FOR I = 1 TO N: READ H(I): NEXT
40  FOR I = 1 TO N: POKE 768,H(I):
50  POKE 769,90: CALL 770
60  NEXT
90  REM *  МАШИННА ПОДПРОГРАМА *
100  DATA 173,48,192,136,208,5,206,
     1,3,240,9,202,208,245,174,
     0,3,76,2,3,96
200  DATA  49: REM БРОЙ НОТИ
290 REM* ВИСОЧИНИНАНОТИ*
300  DATA 255,240,228,215,204,193
     ,185,171,162,153,145,136,128
310  DATA 120,114,107,102,96,92,85,
     81,76,72,68,64
320  DATA 60,56,53,50,48,45,42,40
     ,37,35,33,31
330  DATA 29,28,25,24,23,21,20,19
     ,18,17,16,15
```

7.6. КОМПЮТЪРНО ПИАНО

Програмата 7.10 използува стойностите, определящи височините на музикалните тонове, за да даде възможност за изпълнение на музикални мелодии от клавиатурата на микрокомпютъра. Поради фиксирания брой клавиши обхватът е ограничен на една октава и четири ноти, включително и полутоновете. За преодоляване на това ограничение е предвидена възможност за смяна на височината на клавиатурата. Като се въвеждат числа от 1 до 34, се задава номерът на най-ниската нота в обхвата (основна височина на клавиатурата). От нея нагоре се подреждат останалите 18 ноти. В резултат, колкото по-висока или

по-ниска начална нота се избере, с толкова се измества в съответната посока обхватът на цялата клавиатура. Това не променя музикалния ефект, тъй като нотите запазват честотните отношения помежду си. Съответствието между нотите и клавишите е дадено в табл.7.2.

Таблица 7.2

Клавиш	Нота
A/A	ла^1
B/W	ла^1 # или си^1b
C/S	си^1
Д/D	до^2
Р/R	до^2 # или ре^2b
Ф/F	ре^2
Т/T	ре^2 # или ми^2b
Г/G	ми^2
Х/H	фа^2
У/U	фа^2 # или сол^2b
Й/J	сол^2
И/I	сол^2 # или ла^2b
К/K	ла^2
О/O	ла^2 # или си^2b
Л/L	си^2
+/;	до^3
П/P	до^3 # или ре^3b
Ш/[	ре^3
Щ/]	ми^3

```
1  REM **ПРОГРАМА 7.10**
2  REM **ПИАНО**
10  FOR I = 770 TO 790: READ J: POKE I,J: NEXT
20  READ N: DIM H(N)
30  FOR I = 1 TO N: READ H(I): NEXT
40  HOME : VTAB 10
50  INPUT "ВИСОЧИНА:";W$: IF W$ =
    "" THEN  HOME :END
60 H = VAL (W$): IF H < 1 OR H > 34 THEN 40
70  HOME : VTAB 10: HTAB 13: PRINT
    "ПИАНОТО РАБОТИ";
80  FOR I = 1 TO N - H: H(I) = H(H
    + I - 1): NEXT
90  REM * ПРИЕМАНЕ И АНАЛИЗ НА
     ВЪВЕДЕНИТЕ СИМВОЛИ *
100  GET W$: IF W$ = "A" OR W$ =
    "A" THEN N = 1: GOTO 300
110  IF W$ = "W" OR W$ = "B" THEN
    N = 2: GOTO 300
```

```
120  IF W$ = "S" OR W$ = "C" THEN
     N = 3: GOTO 300
130  IF W$ = "D" OR W$ = "Д" THEN
     N = 4: GOTO 300
140  IF W$ = "R" OR W$ = "P" THEN
     N = 5: GOTO 300
150  IF W$ = "F" OR W$ = "Ф" THEN
     N = 6: GOTO 300
160  IF W$ = "T" OR W$ = "Т" THEN
     N = 7: GOTO 300
170  IF W$ = "G" OR W$ = "Г" THEN
     N = 8: GOTO 300
180  IF W$ = "H" OR W$ = "Х" THEN
     N = 9: GOTO 300
190  IF W$ = "U" OR W$ = "У" THEN
     N = 10: GOTO 300
200  IF W$ = "J" OR W$ = "Й" THEN
     N = 11: GOTO 300
210  IF W$ = "I" OR W$ = "И" THEN
     N = 12: GOTO 300
220  IF W$ = "K" OR W$ = "К" THEN
     N = 13: GOTO 300
230  IF W$ = "O" OR W$ = "О" THEN
     N = 14: GOTO 300
240  IF W$ = "L" OR W$ = "Л" THEN
     N = 15: GOTO 300
250  IF W$ = ";" OR W$ = "+" THEN
     N = 16: GOTO 300
260  IF W$ = "P" OR W$ = "П" THEN
     N = 17: GOTO 300
270  IF W$ = "[" OR W$ = "Ш" THEN
     N = 18: GOTO 300
280  IF W$ = "]" OR W$ = "Щ" THEN
     N = 19: GOTO 300
290  IF W$ = CHR$ (3) THEN  RUN
295  GOTO 100
300  POKE 768,H(N)· POKE 769,64: CALL
     770: GOTO 100
400  DATA 173,48,192,136,208,5,206,
     1,3,240,9,202,208,245,174,
     0,3,76,2,3,96
500  DATA  53: REM БРОЙ НОТИ
550  REM * ВИСОЧИНИ НА НОТИ*
600  DATA 255,240,228,215,204,193
     ,185,171,162,153,145,136,128
610  DATA 120,114,107,102,96,92,8
     5,81,76,72,68,64
```

```
620  DATA 60,56,53,50,48,45,42,40
     ,37,35,33,31
630  DATA 29,28,25,24,23,21,20,19
     ,18,17,16,15
640  DATA 14,13,12,11
```

Ред 400 съдържа машинната подпрограма. Ред 500 съдържа броя на нотите (53), които могат да се изпълнят. Останалите команди DATA съдържат числените стойности, съответстващи на честотите на изпълняваните ноти. С ред 10 се зарежда машинната подпрограма, а в ред 20 се определя размерността на масива H(N), в който се поместват числовите стойности, определящи височините на нотите. С ред 50 се въвежда основната височина на клавиатурата. Тя се приема от програмата, ако е в границите между 1 и 34. С ред 80 се измества индексацията на всички елементи на масива по такъв начин, че числените стойности, определящи височината на нотите, да съответстват на избраната височина на клавиатурата.

От ред 100 започва цикълът, с който се проверяват въведените ноти. В него се сравнява съдържанието на променливата W$ с всички валидни кодове на клавиши за ноти. Когато се открие съвпадение, присвоява се стойност на N, съответстваща на относителната позиция на нотата спрямо началото на избрания обхват (основната височина). С ред 300, към който се изпраща управлението от всички редове на проверяващия цикъл, се изпълнява съответната нота, като N се използва за индексация на масива H(N). Цикълът се повтаря, докато в ред 290 се открие управляващият код MK – Ц/С. Тогава програмата се стартира отново, с което се дава възможност да се зареди нова височина на клавиатурата.

Обхватът се измества с една октава нагоре, като се добави 12 към избраната основна височина. Това следва от обстоятелството, че октавата има 12 полутона.

7.7. ИЗПЪЛНЕНИЕ НА МЕЛОДИИ В ХАРМОНИЯ

Досега ограничихме разглежданията си върху примери, при които в даден момент се изпълнява само една нота. Това ограничение следва от еднопрограмната работа на микрокомпютъра и от начина, по който е съставена машинната програма за произвеждане на музикални звуци. За да осъществим свирене в хармония (изпълнение на повече от една музикална нота в един и същ момент) чрез машинна програма, тя трябва да е значително по-сложна от показаната в табл. 7.1. За да се изпълнят две или повече ноти едновременно, необходимо е да се възпроизведат две или повече честоти и взаимодействието между тях.

Тези трудности се преодоляват, като нотите се изпълняват последователно, но достатъчно бързо, за да ги чуваме едновременно. При този подход на практика се получава известен „вибрато" ефект (тъй като нотите се изпълняват последователно), който става толкова по-забележим, колкото повече ноти трябва да звучат едновременно.

Програмата 7.11 демонстрира използуването на този метод. Тя изпълнява мелодия на три гласа. Нотите от трите гласа се съхраняват в отделни групи елементи на масива M%(36,3). Първоначално нотите на трите гласа са записани като последователност от десетични числа в редове 400, 410 и 420. Чрез редове 50 и 60 те се зареждат в масива M%(36,3) по подходящ начин. Използува се машинната подпрограма от табл.7.1. Тя се съдържа в ред 100 и се зарежда в областта с адреси 770 – 790 с ред 20. По същия начин се зареждат в масив H(N) числовите еквиваленти на 49-те ноти, обхващащи четири октави и един полутон (редове 30,40,200,300–330). Свиренето на мелодията се осъществява от редове 70 и 80. Всеки такт се разделя на пет части, във всяка от които се изпълняват по веднъж нотите на трите гласа. Променливата J брои гласовете от 1 до 3, K индексира частите, на които се разделя тактът, а I брои тактовете.

```
  1  REM **ПРОГРАМА 7.11**
  2  REM **ПЕСЕН В ХАРМОНИЯ**
 10 HA = 768:DA = 769:MPA = 770:D = 10
 20  FOR I = 770 TO 790: READ J: POKE I,J: NEXT
 30  READ N: DIM H(N),M%(36,3)
 40  FOR I = 1 TO N: READ H(I): NEXT
 50  FOR I = 1 TO 36
 60  FOR J = 1 TO 3: READ M%(I,J): NEXT: NEXT
 70  FOR I = 1 TO 36: FOR K = 1 TO 5
 80  FOR J = 1 TO 3: POKE HA,H(M%(I,J)):
     POKE DA,D: CALL MPA: NEXT: NEXT: NEXT
100  DATA 173,48,192,136,208,5,206,
     1,3,240,9,202,208,245,174,
     0,3,76,2,3,96
200  DATA  49: REM БРОЙ НОТИ
290  REM *  ВИСОЧИНИ НА НОТИ *
300  DATA 255,240,228,215,204,193
     ,185,171,162,153,145,136,128
310  DATA 120,114,107,102,96,92,85,
     81,76,72,68,64
320  DATA 60,56,53,50,48,45,42,40
     ,37,35,33,31
330  DATA 29,28,25,24,23,21,20,19
     ,18,17,16,15
```

```
400  DATA 5,8,13,6,10,15,8,13,17,
     13,17,20,15,18,20,13,18,22,13
     17,20,8,13,17,5,8,13,8,12,
     15,8,13,17,8,1,17,8,15,15,5,8,13
410  DATA 3,10,15,6,10,15,5,8,13,
     6,10,15,8,13,17,13,17,20,13,
     17,20,13,18,22,13,17,20,8,13
     ,17,5,8,13,8,12,15,8,13,17,8,13,17
420  DATA 8,13,15,8,12,15,6,8,13,
     3,8,13,1,8,13,1,8,13,1,13,25
     ,1,13,25
```

Прието е всички ноти да имат еднакви продължителности, приблизително равни на 1/4. Отделните ноти се избират чрез индекса на числовия им еквивалент от масива H(N).

От разгледания пример се разбира, че едновременното изпълнение на повече от един глас се програмира подобно на едногласното изпълнение. За целта първо се хармонизира мелодията, т.е. написват се нотите на отделните гласове. След това те се кодират като последователност от числови стойности. Последната се чете от програмата на БЕЙСИК и се подрежда по подходящ начин. За изпълнение на нотите се използува същата машинна подпрограма и числови еквиваленти на музикалните тонове, както при едногласните мелодии. Разликата е, че една и съща нота се изпълнява неколкократно с къса продължителност. Това накъсано изпълнение на нота от един глас се вмества в накъсаното изпълнение на нотите от останалите гласове.

7.8. РЕДАКТОР НА МУЗИКА

В приложената дискета е включен редактор на музика, който се осъществява от програмите ПР 7.12, ПР 7.12А и използува файловете П – ТБЛ и ЗВУК. След като се стартира програмата ПР 7.12, на екрана се извежда менюто на редактора:

```
Редактиране
Пиано
Помощ
Каталог
Край
```

Потребителят избира един от режимите, като придвижва вертикално (с клавишите МК – Х/Н и МК – У/U) светла стрелка и натиска RETURN, когато тя се намира срещу желания режим.

***1*. Редактиране**. В този режим се създават двоични файлове, съдържащи числови стойности, които задават височините и продължителностите на въвежданите в микрокомпютъра тонове (до 200). Въведените тонове се изпълняват веднага и

първоначално се записват в оперативната памет. Оттам могат да се изтриват, променят или запишат върху дискета. След това могат да се четат от дискетата и отново да се зареждат в паметта. Всяка функция се избира с отделен клавиш, действието на който е описано в долната част на екрана.

2. **Пиано.** При този режим върху екрана се изобразяват клавиатура на пиано и петолиние. Клавишите на пианото са означени с букви. При натискане на клавиш микрокомпютърът произвежда тон, чиято височина е равна на височината на означения със същата буква клавиш на пианото и съответната нота се изобразява върху петолинието. По този начин могат да се въведат до 200 ноти, които се изпълняват, след като се натисне RETURN. Освен това те могат да се запишат върху дискета, като текстов файл, откъдето да се четат и изпълняват многократно. Клавишите, с които се извършват съответните операции, са описани в долната част на екрана.

3. **Помощ.** Върху екрана се извеждат числовите стойности, съответствуващи на различните височини и продължителности на нотите. Те са необходими за режим на редактиране.

4. **Каталог.** Действието на този режим е аналогично на действието на на командата CATALOG на ДОС. Чрез него се извеждат имената на файловете от използуваната дискета, без да се излиза от редактора.

От всеки режим се връщаме към основното меню, като натиснем клавиша ОСВ.

ГЛАВА 8. ПРОГРАМИРАНЕ НА ИГРИ

8.1. ОТКРИВАНЕ НА СТЪЛКНОВЕНИЯ

В повечето игри, при които се осъществява движение на фигури, е важно да се регистрира моментът, когато една фигура докосва, застъпва или препокрива друга. Това състояние на взаимното положение на фигурите ще отбелязваме с общото понятие *стълкновение,* а регистрирането му ще наричаме *откриване на стълкновения.* За човека е лесно да види стълкновението на фигурите върху екрана. За компютъра това не е толкова проста задача.

Да си представим, че върху екрана има два обекта. Теоретически е възможно да поддържаме списък на координатите за всяка форма, изобразяваща обект, и да проверяваме за стълкновение, като търсим точки с еднакви координати. Например, ако предположим, че едната форма е разположена върху точки с координати (1,1),(1,2),(1.3), а другата върху (1,2),(2,2),(3,2), те трябва да бъдат в стълкновение, защото точката (1,2) е обща. Тази идея изглежда проста, но на практика е трудноприложима. При осъществяване на анимация непрекъснато променяме точките, които съставят съответната форма. Следователно непрекъснато трябва да променяме и списъка на координатите. При това работата за сравнение на точките по правилото „всяка с всяка" нараства значително с увеличаване на размера на фигурите. Ако проверяваме две прости форми с по 10 точки, ще е необходимо да извършим 100 проверки. За три форми те ще бъдат 1000 и т.н. Очевидно е, че тази идея практически е неприложима.

Ще разгледаме два метода, които използуват по-различен подход. За да улесним обясненията, ще вземем елементи от играта на Дж. Съливан, наречена „Кръстосан огън". В нея се обстрелва цел с управляеми снаряди. Необходимо е да се проследи пътят на снаряда, за да се провери дали той среща целта. Нека въведем таблицата на формите на двата обекта и командите, с които те се зареждат и изчертават:

```
CALL -151
* 6000:02 00 06 00 4A 00 4D 49
* 6008:49 69 18 DF DB DF DB 07
* 6010:48 49 69 4D 49 18 DF DF
* 6018:DB DB 07 48 0D 6D 0D 6D
* 6020:0D D8 DB FB DF DB 48 09
* 6028:0D 0D 0D 4D 01 D8 DF FB
```

```
* 6030:DF FB 08 4D 49 4D 49 05
* 6038:18 DF DB DB DB 07 08 4D
* 6040:49 49 49 05 D8 FB DB DB
* 6048:DF 00 15 3F 2E 35 3F 17
* 6050:4D 31 00
* 3D0G
BSAVE Ф-ТАБЛИЦА,A$6000,L$53
POKE 232,0: POKE 233,96
HGR: HCOLOR = 3: SCALE = 1: ROT = 0
DRAW 1 AT 50,30
DRAW 2 AT 50,80
```

Първата форма е целта. Тя изглежда като паяк. Втората е снарядът.

След като сме въвели таблицата на формите, можем да започнем осъществяването на анимацията. Ще оставим паяка да „виси" в горната част на екрана и ще обърнем повече внимание на движението на снаряда.Можем да го местим наляво и надясно с клавишите Й/J и К/К.Чрез клавиша И/I можем да го изстрелваме нагоре.Той трябва да уцели паяка или да излезе от горната част на екрана.Движението му ще осъществяваме,като го изтриваме и чертаем многократно всеки път с един ред по-високо.Преди всяко преместване ще проверяваме пространството,в което навлиза,за да установим дали там е целта.По този начин вместо да проверяваме дали нещо „удря" целта,ще търсим дали снарядът среща обект по пътя си.

Програмата,която осъществява тази идея,е съставена на два етапа. С поредицата команди, съдържащи се в програма 8.1, се осъществяват основните елементи на анимацията, без да се прави проверка за съвпадение. На екрана се изобразява целта и снарядът се движи под управлението на клавишите Й/J и К/К. Когато натиснем клавиша И/I, компютърът извежда сигнала „звънец", който показва, че тук трябва да се извиква подпрограмата за откриване на стълкновения (тя не е Зключена в програма 8.1).

```
1  REM **ПРОГРАМА 8.1**
2  REM **ОТКРИВАНЕ НА СТЪЛКНОВЕНИЯ**
10 D$ =  CHR$ (13) + CHR$ (4): Z$ = CHR$ (7)
20  PRINT D$"BLOAD Ф-ТАБЛИЦА"
30 XO = 5:X = 5
40  REM *  НАЧАЛНО ИЗЧЕРТАВАНЕ *
85  POKE 232,0: POKE 233,96
90  HGR : HCOLOR = 3: ROT = 0: SCALE = 1
100  DRAW 1 AT 135,50
110  XDRAW 2 AT X,145
130  REM *  ПРОВЕРКА НА КЛАВИШИТЕ *
150  GET W$
```

```
160  IF W$ = "И" OR W$ = "I" THEN
     GOSUB 500: REM ИЗСТРЕЛВАНЕ
200  IF W$ = "Й" OR W$ = "J" THEN
    X = X - 2: GOTO 220
210  IF W$ = "K" OR W$ = "K" THEN
     IF X < 277 THEN X = X + 2
220  IF X < 1 THEN X = 1
230  IF X = XO GOTO 250
240  XDRAW 2 AT XO,145: XDRAW 2 AT
    X ,145:XO = X
250 GOTO 150
500 REM
510 REM  МЯСТО НА ПОДПРОГРАМАТА ЗА
520 REM  ОТКРИВАНЕ НА СТЪЛКНОВЕНИЯ
530 REM
540 PRINT Z$: RETURN
```

С XO е означена хоризонталната координата на снаряда за старото му положение, а с X – за новото. Движението осъществяваме, като променяме X, изтриваме с XDRAW старото положение на снаряда и го начертаваме (отново с XDRAW) в новото (ред 240).

Дотук осъществихме движението на снаряда в долната част на екрана. Сега можем да разгледаме подпрограмата за откриване на стълкновения. Нека си представим, че целта е оградена с малък правоъгълник. Ще го начертаем, като използуваме командите на БЕЙСИК:

```
HPLOT 132,40 TO 148,40
HPLOT TO 148,51
HPLOT TO 132,51
HPLOT TO 132,40
```

Когато движим снаряда нагоре, ще проверяваме дали преминава границите на правоъгълника. Когато ги пресече, това означава, че трябва да изпълним частта от програмата,имитираща експлозията. Тази идея е осъществена в програма 8.2.

```
1  REM * * ПРОГРАМА 8.2 * *
2  REM * * ОТКРИВАНЕ НА СТЪЛКНОВЕНИЕ
       ЧРЕЗ ОБГРАЖДАЩ ПРАВОЪГЪЛНИК * *
50  REM * ГРАНИЦИ НА ПРАВОЪГЪЛНИКА *
60 XN = 132:XX = 148:YMAX = 51
500  REM * ОТКРИВАНЕ НА СТЪЛКНОВЕНИЯ *
510  PRINT Z$
520 C = 0: REM ФЛАГ ЗА СТЪЛКНОВЕНИЕ
530  FOR Y = 144 TO 0 STEP - 1
540  IF Y < YMAX THEN  IF X > XN AND
    X < XX THEN C = 1
```

```
550  REM * СТЪЛКНОВЕНИЕ *
560  IF C THEN PRINT  Z$;Z$;Z$;Z$
     : XDRAW 2 AT X,Y + 1: GOTO 630
570  XDRAW 2 AT X,Y + 1: XDRAW 2 AT X,Y
580  NEXT Y
590  REM * СНАРЯДЪТ ИЗЛИЗА ОТ ЕКРАНА *
600 Y = Y + 1
610  XDRAW 2 AT X,Y
620  REM * НОВ СНАРЯД *
630  XDRAW 2 AT X,145
640  RETURN
```

В ред 60 се установяват границите на правоъгълника. Горната граница е пропусната, защото в разглеждания пример снарядът идва винаги отдолу.

Стълкновенията се откриват в ред 540. Най-напред се проверява дали координатата Y на върха на снаряда е достигнала долната граница на правоъгълника. Ако това е вярно, се проверява дали координатата X е между установените минимална и максимална граница. При изпълнение на двете условия се вдига флагът за стълкновение (C = 1).

Ред 560 е мястото, откъдето се извършва преход към подпрограмата, имитираща експлозията. В нашия пример при експлозия изпълняваме неколкократно сигнала „звънец". След това изтриваме снаряда от мястото, в което се намира, и преминаваме към ред 630, за да начертаем нов.

Тъй като целта е неподвижна, целесъобразно е първо да проверяваме координатата X, а след това – Y. Ако X е в границите на целта, можем да очакваме стълкновение. Само тогава е необходимо да проверяваме Y. В обратния случай можем да оставим снаряда да се движи, без да извършваме проверка.

Методът с обграждащ правоъгълник е сравнително прост, но не е достатъчно точен. При него стълкновението се открива чрез въображаем правоъгълник, а не вследствие на действително препокриване на фигурите. Освен това се проверява само по централната ос на снаряда, докато краищата му също могат да засегнат целта.

Проверката по цялата широчина на снаряда не е трудна и ще оставим да я направите сами. По-интересен е въпросът за откриване на непосредствено застъпване между фигурите. Той може да се реши, като се проверява клетката от паметта, управляваща точката от екрана, в която навлиза снарядът. Ако тя е активизирана, т.е. управляващият я бит е 1, ще настъпи стълкновение. В примера единствените активизирани точки от екрана са тези на целта.

Въпреки че проверката на отделни битове от паметта със средствата на БЕЙСИК е неудобна, ще демонстрираме тази идея чрез програма 8.3. Тя променя и допълва програма 8.2.

```
1  REM **ПРОГРАМА 8.3**
2  REM **ОТКРИВАНЕ НА СТЪЛКНОВЕНИЕ
       ЧРЕЗ ПРОВЕРКА ЗА ПРЕПОКРИВАНЕ**
5  GOSUB 1000
500    REM *ОТКРИВАНЕ НА СТЪЛКНОВЕНИЯ*
510  PRINT Z$
515 CH% = X / 7:O% = X - 7 * CH%: O% = O% + 1
520 C = 0
530  FOR Y = 144 TO 0 STEP  - 1
536 B% =  PEEK (Y%(Y) + CH%)
538  FOR I = 7 TO O% STEP  - 1
540 S% = 2 ^ I
542  IF B% >  = S% THEN  B% = B% - S%
544  NEXT I
546  IF B% >  = 2 ^ I THEN C = 1
550  REM *ОТКРИВАНЕ НА СТЪЛКНОВЕНИЯ*
560  IF C THEN PRINT Z$;Z$;Z$;Z$
    : XDRAW 2 AT X,Y + 1: GOTO 630
570  XDRAW 2 AT X,Y - 1: XDRAW 2 AT X,Y
580  NEXT Y
590  REM *СНАРЯДЪТ ИЗЛИЗА ОТ ЕКРАНА*
600 Y = Y + 1
610  XDRAW 2 AT X,Y
620  REM *НОВ СНАРЯД*
630  XDRAW 2 AT X,145
640  RETURN
1000  REM *ИЗЧИСЛЯВАНЕ НА Y-КООРДИНАТИТЕ*
1010  DIM Y%(192)
1020  FOR I = 0 TO 184 STEP 8: READ NA%
1030  FOR J = 0 TO 7: Y%(I + J) = NA% + J * 1024
1040  NEXT J,I
1100  DATA 8192,8320,8448,8576,8704,
     8832,8960,9088
1110  DATA 8232,8360,8488,8616,8744,
     8872,9000,9128
1120  DATA 8272,8400,8528,8656,8784,
     8912,9040,9168
1130  RETURN
```

Предназначението на подпрограмата, започваща от ред 1000, вече е известно. С нея се получава таблица, която съдържа началните адреси на всеки ред от екрана.

В ред 515 координатата X се разделя на 7, за да се определи кой от 40-те байта на реда (CH%) съдържа бита, управляващ точката от екрана, към която настъпва снарядът. После се определя номерът на бита (O%) вътре в байта. Ако предположим, че снарядът е изстрелян с хоризонтална координата X = 52,

върхът му се намира в точка номер 3 (започваме да броим от 0) на байт 7. Това означава, че преди всяко преместване на снаряда нагоре трябва да проверяваме третата точка от полето непосредствено над него. Докато тя не е активизирана, т.е. докато управляващият я бит не е станал 1, снарядът няма да срещне обект по пътя си. Ако е активизирана, снарядът е непосредствено пред целта и трябва да вдигнем флага за стълкновение.

Проверката на отделна точка от екрана се осъществява от редове 538 – 546. Преди това в ред 536 променливата B% се зарежда със съдържанието на байта, който се проверява. За да обясним действието на командите в тези редове, ще предположим, че B% е заредена примерно с двоичната стойност 00111011 (59) и ще проследим действието им стъпка по стъпка.

Ако хоризонталната координата X е 52, в ред 515 се изчислява O% = 3. От това следва, че ще се проверява третата точка от полето на екрана, управлявано от съответния байт. Тъй като точката с номер 3 съответствува на четвъртия бит в байта отдясно наляво, към O% се прибавя 1 и се влиза в цикъла, започващ от ред 538.

Променливата S% приема стойностите на степените на числото 2 при промяна на степенния показател I. Тези стойности съответствуват на теглото на разредите в двоичните числа. Например 2 ^ 7 = 128 е теглото на осмия бит от байта, 2 ^ 6 = 64 – на седмия бит и т.н.

При всяко изпълнение на цикъла се проверява отляво надясно по един бит от байта, като той се установява в 0, ако е бил 1. Това продължава, докато се стигне до бита, чийто номер се съдържа в O%.

Проследете тези действия, като използувате таблицата:

I	S%	B%
7	128	59 = 00111011
6	64	59 = 00111011
5	32	27 = 00011011
4	16	11 = 00001011
3	–	11 = 00001011

В резултат B% е равно на 00001011 (11). Нашата цел е да проверим четвъртия бит. Това се извършва от ред 546. В него се проверява дали B% е равно или по-голямо от 2 ^ I. За разглеждания случай при излизане от цикъла I = 3. Ако стойността на B% е по-голяма от 2 ^ 3, четвъртият бит ще бъде 1, тъй като всички по-старши битове вече са нулирани. В този случай флагът за стълкновение се вдига и задачата за проверка на четвър-

тия бит е изпълнена. Тя ни струваше доста усилия, но още в началото отбелязахме, че със средствата на БЕЙСИК се решава трудно.

В следващата таблица се проследява изпълнението на същата последователност от операции за B% = 50, при което четвъртият бит е нула:

I	S%	B%
7	128	50 = 00110010
6	64	50 = 00110010
5	32	18 = 00010010
4	16	2 = 00000010
3	–	2 = 00000010

Тук флагът за стълкновение няма да се вдигне, защото условието B% > = 2 ^ 3 (ред 546) не се изпълнява.

Недостатък на току-що разгледаната програма (както и на предишната) е, че проверката се извършва само за централната ос на снаряда. Той може да засегне целта с левия или с десния си край, но стълкновение няма да се регистрира. С прости допълнения може да се реализира проверка по цялата широчина. Тях пак ще ги оставим на вашето въображение.

Предимство на последния вариант на програмата е регистрирането на стълкновение само при действително застъпване на обектите. То обаче е за сметка на забавяне на анимацията, тъй като описаните по-сложни проверки изискват повече време. С някои подобрения на програмата на БЕЙСИК може да се постигне известно ускорение на процеса, но действително добри резултати се получават с програма на асемблерен език.

Откриването на стълкновения по описаните методи се усложнява, ако върху екрана има допълнителни обекти, например облаци и др. Застъпването на снаряда с тях не трябва да се взема предвид. Тези затруднения могат да се преодолеят по два начина.

Първият предвижда поддържане на таблица за текущото положение на целите. Когато флагът за стълкновение се вдигне, таблицата се претърсва, за да се провери дали има цел на това място. Ако има, тя се разрушава, ако няма – флагът за стълкновение се сваля.

При втория начин се използуват двете графични страници на режим BPC. В едната се поддържат изображенията на всички фигури, които се появяват на екрана. Другата съдържа само снаряда и целите. Тя не се извежда, а се използува от подпрограмата за проверка на стълкновения. Разбира се, поддържането на изображения в двете страници едновременно изисква допълни-

телно време. Това често е причина да прибегнем отново до асемблерния език.

Когато движещите се фигури се чертаят с командите DRAW и XDRAW, има по-лесна възможност за откриване на стълкновения. Проверява се съдържанието на клетка 234 и по него се съди за наличието или отсъствието на стълкновение, като се знаят следните особености.

Когато се изпълнява командата DRAW, тя записва винаги в клетка 234 броя на точките от полето, върху което чертае, чийто цвят съвпада с цвета на чертане в момента. Когато се изпълнява XDRAW, тя записва в адрес 234 броя на действително изчертаните точки (при изпълнение на XDRAW може да се чертае или да се изтрива). Например, ако с командата DRAW се чертае фигура, която е начертана преди това със същия цвят, в клетка 234 ще се запише число, показващо броя на застъпващите се точки. Ако с XDRAW се чертае върху фигура, създадена с XDRAW, в резултат от изпълнението ѝ в клетка 234 ще се запише броят на незастъпващите се точки, тъй като в този случай ще се изчертаят само те.

Преди всяко изпълнение на командите DRAW и XDRAW клетка 234 се нулира автоматично. Поради това, ако при чертане с DRAW няма застъпващи се точки с еднакви цветове или при чертане с XDRAW има пълно препокриване, съдържанието на клетка 234 остава 0.

Възможността за откриване на стълкновения, като се проверява адрес 234, е показана в програма 8.4. Тя използува същите фигури, между които се търси стълкновение, поради което в първата си част (редове 10 – 250) е подобна на програма 8.1.

```
1  REM **ПРОГРАМА 8.4**
2  REM **ОТКРИВАНЕ НА СТЪЛКНОВЕНИЯ
     ЧРЕЗ ПРОВЕРКА НА КЛЕТКА 234**
10 D$ = CHR$ (13) + CHR$ (4):Z$
     = CHR$ (7)
20  PRINT D$"BLOAD Ф-ТАБЛИЦА"
30 XO = 5:X = 5: HGR
50  HCOLOR= 3: ROT= 0: SCALE= 1
60  POKE 232,0: POKE 233,96
70  REM *НАЧАЛНО ИЗЧЕРТАВАНЕ*
100  DRAW 1 AT 135,50
110  XDRAW 2 AT X,145
130  REM *ПРОВЕРКА НА КЛАВИШИТЕ*
150  GET W$
160  IF W$ = "И" OR W$ = "I" THEN
       GOSUB 500: REM ИЗСТРЕЛВАНЕ
200  IF W$ = "Й" OR W$ = "J" THEN
     X = X - 2: GOTO 220
```

```
210  IF W$ = "K" OR W$ = "K" THEN
     IF X < 277 THEN X = X + 2
220  IF X < 1 THEN X = 1
230  IF X = XO GOTO 250
240  XDRAW 2 AT XO,145: XDRAW 2 AT
     X,145:XO = X
250  GOTO 150
500  REM * ОТКРИВАНЕ НА СТЪЛКНОВЕНИЕ *
520  FOR Y = 144 TO 1 STEP  - 1
540  XDRAW 2 AT X,Y + 1.
550  IF  PEEK (234) <  > 0 THEN GOTO 600
555  XDRAW 2 AT X,Y
560  NEXT Y
570  REM * НЯМА СТЪЛКНОВЕНИЕ *
580 Y = Y + 1: XDRAW 2 AT X,Y
590  XDRAW 2 AT X,145: RETURN
600  REM * ИМА СТЪЛКНОВЕНИЕ *
610  PRINT Z$;Z$;Z$;Z$: XDRAW 2 AT X,145
620  RETURN
```

Проверката за стълкновения се извършва от ред 550. Ако при изпълнението му съдържанието на клетка 234 е равно на нула, няма стълкновение. Това е така, защото с командата XDRAW в ред 540 се изтриват старите положения на снаряда. Ако той е начертан върху черно поле (не върху друг обект), в резултат на изтриващото изпълнение на XDRAW клетка 234 ще се зареди с нула. Във всеки друг случай, когато първото начертаване на снаряда с XDRAW (ред 555) е застъпило някакъв обект, изтриващото изпълнение на XDRAW ще запише в клетка 234 стойност, различна от нула. Тогава управлението се изпраща към ред 600, където се изпълняват командите, имитиращи последствията от стълкновението.

8.2. ПРОГРАМИ ЗА ИГРИ

Както всеки клас програми, предназначени за решаване на определена група задачи, така и програмите за игри се характеризират със специфични особености. Те съдържат елементи, които повече или по-малко са необходима част от всяка програма и изпълняват аналогични функции. Основните блокове на програмите за игри в повечето случаи трябва да осъществяват:

* установяване на началните условия на играта;

зареждане или рисуване на статичната картина на екрана, която е постоянна през цялата игра или през определени етапи от нея ;

зареждане или съставяне таблици на форми;

извеждане на текст за надписи или резултати от играта, най-често в режим BPC;

звукови ефекти и музика;
анимация на фигури;
подходящо оцветяване на изображението (статично и подвижно);
проверка за стълкновения;
* изчисляване на резултата от играта;
приемане на команди от контролерите за игри или клавиатурата;
* общата логика (алгоритъмът) на играта.

Програмистите с по-дълъг опит ще забележат, че само три (отбелязани със *) от изброените 11 функции могат да се изпълнят с програмните средства на езиците от високо ниво, които бяха на разположение преди 6 или 5 години. Осъществяването на останалите функции изисква знания, похвати и програмни средства, специфични за програмирането на графични изображения, музика и игри. Изложеният материал дотук засяга тези въпроси .Затова книгата може да ви помогне да преодолеете много от трудностите, свързани със създаването на програми за игри. За целта е необходимо да я прочетете внимателно и да приложите наученото с достатъчно фантазия.

Описани са съдържащи се в приложената дискета програмни инструменти, използуването на които облекчава и ускорява процеса на създаване на програми за игри. Те могат да се използуват от недостатъчно квалифицирани програмисти за създаване на отделни програмни блокове, без да е необходимо да се усвояват тънкостите, свързани с тяхното програмиране.

Чрез графичните редактори за HPC или BPC може да се начертаят предварително статичните картини на изображението и да се съхранят върху магнитен носител. След това не е трудно да се зареждат от програмата на играта винаги когато тя се изпълнява.

С музикалния редактор могат да се създадат звукови ефекти и мелодии, които да се използуват впоследствие по същия начин. Редакторът за форми и таблици на форми предоставя широки възможности за създаване на статични и движещи се фигури. Освен това с него могат да се създадат различни по размер и шрифт текстови символи. Всеки от тези „полуфабрикати" може да се съхрани в двоичен, текстов или изпълним файл и при необходимост да се зарежда от програмата на играта.

8.3. ИГРИ

Всеки обича компютърните игри. Те доставят удоволствие и превръщат компютъра в забавен партньор. Освен приятно прекараното време и отмората те носят и полза. Учат ни да мислим стратегически, да планираме бъдещето. Внасят ред в

мислите и развиват рефлексите. Те са пътят за сприятеляване с компютъра, за да бъде приет като ежедневен помощник.

В приложената дискета са записани две програми на игри — Приземяване и Френска стратегическа игра. Първата от тях представя категорията игри, които бързо се разпространиха през последните години. Съществена част в тях заемат графиката и анимацията. Втората игра е от логически тип. Нейният алгоритъм е известен отдавна. Тук тя е представена във версията си за Правец-82.

Приземяване

А. Условия на играта

Играчът управлява приземяването на извънземни същества. Той има възможност да ги придвижва в пет посоки. Целта е да се приземят колкото може повече „извънземни", преди да свърши горивото. Количеството гориво се изобразява графически в долната част на екрана.

Играта започва с появата на „извънземното" над ръба на скала. Играчът трябва да го отдели оттам и да управлява падането му чрез допълнителни тласъци, като го насочва към някоя от шестте цилиндрични площадки. Тласъците в петте посоки се задават, като се натиска съответен бутон от клавиатурата, но костват разход на гориво. Приземяването с висока скорост води до катастрофа.

Играчът трябва да приземи колкото може повече фигури върху една площадка, но не трябва да ги качва една върху друга. В този случай те се разрушават.

Спечелените точки зависят от размера на площадката и скоростта на приземяване. По-малката площадка и по-плавното приземяване се оценяват по-високо.

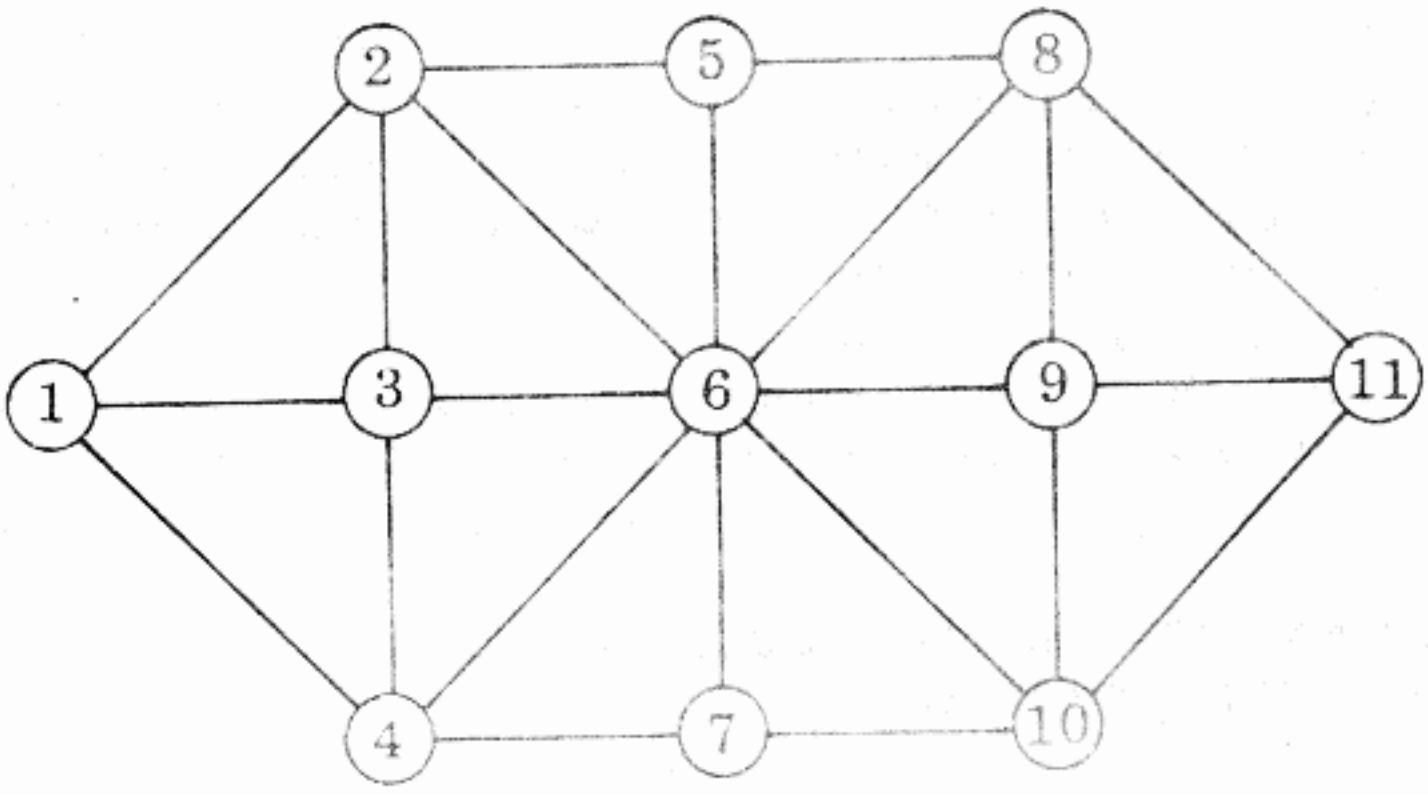
Играчът губи точки при неправилно приземяване. Ако успее да набере 4000 точки преди края на играта, получава еднократно допълнително гориво.

Б. Предварителна подготовка

В приложената дискета програмата на играта е означена с името ПР 8.6 и се стартира с командата

RUN ПР 8.6

Ако се зарежда от дискета, в която липсват файловете ТАБЛИЦА НА СИМВОЛИ и ТБЛ-ПРЗ, необходимо е преди стартирането ѝ да се изпълнят програмите ПР 5.3 и ПР 8.5. Те създават таблици на форми и ги записват върху дискета. Оттам програмата на играта ги зарежда в паметта при всяко стартиране и ги използува. Първата програма създава таблица на формите на текстовите символи и я записва под името ТАБЛИЦА НА СИМВОЛИ. Тя се използува за извеждане на надписи и резултати от играта върху графичен екран (за повече информация вж. гл.5). Втората програма създава таблица на форми, които играта използува за изобразяване на движещи се фигури. Тя се записва с името „ТБЛ-ПРЗ".



Фиг. 8.1

В. Управление на играта

нова игра се започва с клавиша RETURN;

играта се завършва с клавиша ОСВ;

клавишите И/I, О/O, К/K, Й/J и У/U придвижват приземяваната фигура съответно наляво, наляво и нагоре (по диагонал), нагоре, нагоре и надясно (по диагонал) и надясно.

Френска стратегическа игра

А. Условия на играта

Тази игра е описана от Едуард Люк в книгата му „Математически развлечения". По-късно е анализирана от Мартин Гарднер в „Математически игри". Тя съчетава прости правила и добре уточнена стратегия.

Схемата на играта е дадена на фиг.8.1. Играят двама противници, които движат последователно фигурите си в номерирани полета. Преместването от едно поле към друго е възможно само ако са свързани с линия. Единият играч има три бели фигури, които в началото на играта са разположени в полета 1,2 и 4. Другият има само една черна фигура първоначално разположена в поле 6.

Първият ход е на белите. Те могат да се движат напред, наляво, надясно и по диагонал, но само в посока към поле 11 (не могат да се връщат). Например от поле 2 бялата фигура може да се премести към полета 3, 5 или 6 (ако са свободни), но не може да се върне към поле 1. Целта на играча с белите фигури е да загради черната фигура по такъв начин, че другият играч да не може да направи следващ ход. Тогава белите печелят. Играчът с черната фигура печели, когато стигне до поле 1 или белите фигури повторят 20 пъти един и същ ход.

Б. Стартиране на играта

В дискетата програмата на играта е означена с името ПР 8.7 и се стартира с командата

RUN ПР 8.7

В. Управление на играта

Играещият срещу компютъра задава номерата на полетата, в които мести фигурите си, като отговаря на въпросите:

Вашият ход: от позиция
на позиция

Клавишът ОСВ се използва за преминаване от текстовото към графичното изображение на играта.

ПРИЛОЖЕНИЕ 1

ТАБЛИЦА НА УПРАВЛЯВАЩИ КОДОВЕ И КОДОВЕ НА СИМВОЛИ В ПРАВЕЦ-82

УПРАВЛЯВАЩИ КОДОВЕ

Дес.	Шестн.	Означение в ASCII	Клавиши	Значение в ASCII
0	0	NULL	МК – Ю/@	Нулев код
1	1	SOH	МК – А/A	Начало на заглавие
2	2	STX	МК – Б/B	Начало на текст
3	3	ETX	МК – Ц/C	Край на текст
4	4	ET	МК – Д/D	Край на предаване
5	5	ENQ	МК – Е/E	Запитване
6	6	ACK	МК – Ф/F	Потвърждение
7	7	BEL	МК – Г/G	Звънец
8	8	BS	МК – Х/H	Една позиция назад
9	9	HT	МК – И/I	Хоризонтална табулация
10	A	LF	МК – Й/J	Нов ред
11	B	VT	МК – К/K	Вертикална табулация
12	C	FF	МК – Л/L	Нова страница
13	D	CR	МК – М/M	Начало на ред
14	E	SO	МК – Н/N	Долен регистър
15	F	SI	МК – О/O	Горен регистър
16	10	DLE	МК – П/P	Освобождаване на линия
17	11	DC1	МК – Я/Q	Управление на устройство 1
18	12	DC2	МК – Р/R	Управление на устройство 2
19	13	DC3	МК – С/S	Управление на устройство 3
20	14	DC4	МК – Т/T	Управление на устройство 4
21	15	NAK	МК – У/U	Отказ
22	16	SYN	МК – Ж/V	Синхронизация
23	17	ETB	МК – В/W	Край на блок
24	18	CAN	МК – Ь/X	Анулиране
25	19	EM	МК – Ъ/Y	Край на носител
26	1A	SUB	МК – З/Z	Заместване
27	1B	ESC	ОСВ/МК – Ш/[	Освобождаване
28	1C	FS	МК – Э/\	Разделител на файлове
29	1D	GS	МК – Щ/]	Разделител на групи
30	1E	RS	МК – Ч/^	Разделител на записи
31	1F	US	Недостъпен	Разделител на елементи

КОДОВЕ НА СИМВОЛИ

(БУКВИ, ЦИФРИ И СПЕЦИАЛНИ ЗНАЦИ)

Дес.	Шестн.	Символ	Дес.	Шестн.	Символ	Дес.	Шестн.	Символ
32	20	Интервал	64	40	Ю	96	60	@
33	21	!	65	41	А	97	61	А
34	22	"	66	42	В	98	62	Б
35	23	#	67	43	С	99	63	Ц
36	24	$	68	44	D	100	64	Д
37	25	%	69	45	E	101	65	Е
38	26	&	70	46	F	102	66	Ф
39	27	'	71	47	G	103	67	Г
40	28	(	72	48	H	104	68	Х
41	29	)	73	49	I	105	69	И
42	2A	*	74	4A	J	106	6A	Й
43	2B	+	75	4B	K	107	6B	К
44	2C	,	76	4C	L	108	6C	Л
45	2D	-	77	4D	M	109	6D	М
46	2E	.	78	4E	N	110	6E	Н
47	2F	/	79	AF	O	111	6F	О
48	30	0	80	50	P	112	70	П
49	31	1	81	51	Q	113	71	Я
50	32	2	82	52	R	114	72	Р
51	33	3	83	53	S	115	73	С
52	34	4	84	54	T	116	74	Т
53	35	5	85	55	U	117	75	У
54	36	6	86	56	V	118	76	Ж
55	37	7	87	57	W	119	77	В
56	38	8	88	58	X	120	78	Ь
57	39	9	89	59	Y	121	79	Ъ
58	3A	:	90	5A	Z	122	7A	З
59	3B	;	91	5B	Ш	123	7B	[
60	3C	<	92	5C	\	124	7C	Э
61	3D	=	93	5D	Щ	125	7D	]
62	3E	>	94	5E	Ч	126	7E	Ю
63	3F	?	95	5F	_	127	7F	DEL

СЪДЪРЖАНИЕ

ОЧАКВАЙТЕ:

КНИГА 6. ЩО Е ДИСКОВА ОПЕРАЦИОННА СИСТЕМА

ОРЛИН ВЪЛЧЕВ

На достъпен език е описана дисковата операционна система ДОС 3.3, с която работи персоналният компютър Правец-82. Дадени са сведения за запомнящите устройства с гъвкав и твърд магнитен диск. Разгледана е структурата на записите върху дискетата. Показана е работата с различните видове файлове. Подробно са дадени командите на дисковата операционна система. Показано е как са зарежда системата. В приложения са дадени командите и съобщенията на ДОС. Включени са примери, упътвания и препоръки.

Книгата е ориентирана към всички, които работят с компютъра Правец-82 и искат да се научат да го използуват по-пълноценно.

КНИГА 7.50 ПРОГРАМИ ЗА ПЕРСОНАЛЕН КОМПЮТЪР

ТОДОР ЕВТИМОВ, АЛЕКСАНДЪР ОМАЙСКИ И ГЕОРГИ ТОШКОВ

Програмите в книгата са посветени на организирането на диалога с програмата, създаването на структури от данни и достъпа до тях, оформянето на резултатите от изпълнението на програмите, методите за сортиране на данни. Предложен е набор от помощни програми и сервизни процедури. Програмите са написани на БЕЙСИК за персонален компютър Правец-82, текстът им е отпечатан и коментиран в книгата, а те са включени в приложената към нея дискета.

Книгата е предназначена за читатели, запознати с персоналния компютър Правец-82 и преминали начално обучение по програмиране на БЕЙСИК.

Към „оредицата „Микрокомпютърна техника за всички" се разпространяват дискети към следните книги:

3. БЕЙСИК — език на персоналните компютри
   Атанас Шишков и Татяна Бояджиева
7. 50 програми за персонален компютър
   Тодор Евтимов, Александър Омайски, Георги Тошков
8. ПАСКАЛ за персонални компютри
   Моско Аладжем
9. Компютърът играе, рисува и свири
   Велчо Велев и Надежда Петрова

МИКРОКОМПЮТЪРНА ТЕХНИКА ЗА ВСИЧКИ. Книга 9.
КОМПЮТЪРЪТ ИГРАЕ, РИСУВА И СВИРИ

АВТОРИ: к.т.н. инж. ВЕЛЧО СТОЙНОВ ВЕЛЕВ
к.т.н. инж. НАДЕЖДА КРЪСТЕВА ПЕТРОВА

РЕЦЕНЗЕНТИ: ст.н.с. к.т.н. инж. ВЛАДИМИР РУСЛАНОВ ЧИЛОВ
инж. АЛЕКСИ СТЕФАНОВ БОЮКЛИЕВ

Националност българска
Първо издание

Код 9533112231 / 3205—8—87
Изд. № 15030

Научен редактор инж. СНЕЖИНА БАДЖЕВА
Художник ДОСЮ ДОСЕВ
Художествен редактор МАРИЯ ДИМИТРОВА
Технически редактор ЦВЕТАНА ПОПОВСКА
Коректор ЕКАТЕРИНА ЕВСТАТИЕВА

Дадена за набор на
Подписана за печат м. декември 1986 г.
Излязла от печат м. януари 1987 г.
Формат 60 × 90/16
Печ. коли 9
Изд. коли 9
УИК 9,36
Тираж 100 000 + 106
Цена 1,10 лв.

Държавно издателство „Техника", бул. Руски 6 — София
Държавна печатница „Георги Димитров" — София

МИКРОКОМПЮТЪРНА ТЕХНИКА ЗА ВСИЧКИ

1. РАБОТА С ПЕРСОНАЛЕН КОМПЮТЪР
2. ПРОГРАМИРАНЕТО — И ПРОСТО, И СЛОЖНО
3. БЕЙСИК — ЕЗИК НА ПЕРСОНАЛНИТЕ КОМПЮТРИ
4. МИКРОПРОЦЕСОРЪТ — СЪРЦЕТО НА МИКРОКОМПЮТЪРА
5. КАК РАБОТИ ПРАВЕЦ-82
6. ЩО Е ДИСКОВА ОПЕРАЦИОННА СИСТЕМА
7. 50 ПРОГРАМИ ЗА ПЕРСОНАЛЕН КОМПЮТЪР
8. ПАСКАЛ ЗА ПЕРСОНАЛНИ КОМПЮТРИ
9. КОМПЮТЪРЪТ ИГРАЕ, РИСУВА И СВИРИ
10. ПЕРИФЕРНИ УСТРОЙСТВА ЗА ПЕРСОНАЛНИ КОМПЮТРИ
11. ПРОФЕСИОНАЛНИ КОМПЮТРИ
12. НАПРАВЕТЕ САМИ МИКРОКОМПЮТЪР

ИЗДАТЕЛСТВО ТЕХНИКА

1,10