16

КС4

КООРДИНАТАТА Y

Поредният тест е естествено продължение на темата от миналия. След като сме се справили с X-координатата, защо да не се справим и с Y? Тук Y-координатата е само един байт, затова пък адресът на графичната страница е два байта. На пръв поглед адресите на началата на 192-та (десетично) реда на графичната страница изглеждат доста разбъркани и любителите на таблиците и високите скорости с удоволствие биха похарчили за целта 192*2 байта. Е, така всеки може. Ами ако няма толкова свободни байтове за програмата? Опитайте, при условие, че Y-координатата е в акумулатора, да запишете в два байта в нулевата страница адреса на най-левия байт, изобразяван на екрана в графичен режим 280x192 точки (в коя от двете графични страници е въпрос на вкус), който съответства на тази Y-координата. Имате на разположение и два работни байта в нулевата страница. Решението ви заедно с таблиците не трябва да надхвърля 36 байта и да не се изпълнява за повече от 54 такта на 6502. Приятна работа!

АВГУСТ СТОЯНОВ

82

Решение на КС3

16

ДЕЛЕНИЕ НА СЕДЕМ

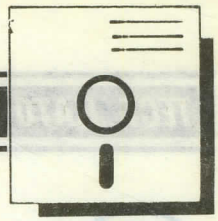
(Великолепната седморка)

от АВГУСТ СТОЯНОВ

Тъй като технологията на издаване на списание КВ е такава, че когато отговорът на теста трябва да бъде предаден за печат, броят с условието още не е излязъл, авторът няма представа доколко проблемът в теста е предизвикал интерес и какъв процент от вас са се справили с него, колко са допуснали грешки или са отишли далеч от истината. Затова предварително моля за извинение онези, за които проблемът и решението не се нуждаят от коментар. Разбира се, ако вие, уважаеми читатели, сметнете, че коментарът към условията или решенията по принцип е излишен, то те ще се отпечатват без него. Предполагайки, че досегашните задачи от теста „Като Слънце“ бяха твърде лесни за някои от вас, тази вече е с повишена трудност. Тъй като преценката за трудността (за разлика от ефективността) е субективна, приемам за съвсем нормално, че ще има читатели, чиято преценка се отличава катастрофално от моята. И това е много вероятно — малко преди да предам решението в редакцията, открих една излишна инструкция и веднага я изхвърлих. Така че решението вече заема 48 байта и се изпълнява за 69 такта на 6502! Но много е възможно да се намери и по-кратко решение.

Решението на тази задача е илюстрация на един от парадоксите (за непишещите на асемблер) в програмирането, при които в конкретен случай по-елегантното (и по-ефективно) решение не е реализация на елегантен алгоритъм. Съществено улеснение в случая се оказва това, че делимото (x-координатата) е по-малко от 512, т. е. дължината му е само 9 бита. За по-голяма нагледност при обяснението въвеждам следните означения.

Разделям използваните числа на битови триади, като всяка триада се означава с главна латинска буква; всяко число се представя като последователност от триади (една, две или три в зависимост от големината си), означени с последователни букви от азбуката, като по-предна буква означава по-младша триада, напр. CBA, ZYX, MLK; нулевата триада се означава с 0;



малките букви означават променливи; аритметичните знакове имат общоприетото значение, с изключение на знаковете за инфиксни операции целочислено деление и остатък по модул, означени с "/" и "!".

Нека преминем към проблема. Ако решим просто да делим 9-битово число на 3-битово, ще се убедим колко много операции за изместване, събиране и изваждане ще ни трябват, като някои от тях се прилагат върху 9-битови числа. Може би, написано в цикъл, решението, използващо стандартен алгоритъм за делене, да заеме по-малко място, но със сигурност ще бъде много по-бавно. Още по-късо изглежда деленето с изваждане (използвано в APPLESOFT BASIC), но то е на порядъци по-бавно. Една „по-прогресивна" идея е да се използва умножаване по една седма. В двоично представяне една седма изглежда като 0.001001001.... След умножаването числото CBA би изглеждало така:

```
  CB.A
+ C.BA
+ 0.CBA
+ 0.0CBA
+ ..... и т. н.
```

Вижда се, че всяка цифра на сумата след десетичната точка е (A+B+C)!8+(A+B+C)/8 и всяка крайна сума е с недостиг. Искаме да определим стойността на цялата част на тази сума, защото това е частното при целочисленото деление на 7. За да не смятаме излишно точно, достатъчно е да добавим (сумата е с недостиг) към сумата най-голямото число от вида 1/7, по-малко от 1/7, тъй като умножаването му по 7 няма да се отрази на резултата. Това число е 1/8, т. е. трябва да добавим 1 към първата триада след точката. И тъй като тази триада трябва да е точна с недостиг, към нея трябва да добавим и преноса от следващата или от същата, понеже съвпадат. И така частното от делението на числото CBA на 7 е

CB+C+[A+B+C+1+(A+B+C)/8]/8.

Вижда се, че тук се използват само събирания, целочислено делене на 8 (изместване) и вземане по модул 8 (and#7), за да се получат отделните цифри A, B и C. Сега да получим остатъка. Общата формула x=7*q+r, където x е делимото, q е частното, а r – остатъкът, можем да представим като x=(8*q+r)–q. Вземайки двете страни на равенството по модул 8 (т. е. само младшите им триади), получаваме x!8=(r–q!8)!8, тъй като 8*q+r има същия остатък по модул 8 както r (очевидно е, че младшата триада на 8*q е 0, а r<8). От това равенство получаваме r=(x!8+q!8)!8, или r=(x+q)!8. Това означава, че знаейки частното, получаваме остатъка само с едно събиране с делимото и един and #7. Очевидно този начин е значително по-прост от изваждането от делимото на частното, умножено по 7. Въпреки елегантното намиране на остатъка и сносното намиране на частното бихте могли да се убедите, че на 6502 такова решение заема повече от 48 байта и се изпълнява за повече от 69 такта. Къде е проблемът? Явно в частното. Грубото му намиране става само с едно събиране CB+C, а уточняването изисква сума операции. Дали не можем да намерим грубо и частното, и остатъка и да ги уточняваме едновременно? Нека вземем за грубо частно сумата q=CB++C+(A+B+C)/8. Използвайки формулата за остатъка r=x–q*8+q, ще получим последователно:

r=CBA–[CB+C+(A+B+C)/8]*8+CB+C+(A+B+C)/8,
r=CBA–CB0–C0–A–B––C+(A+B+C)!8+CB+C++(A+B+C)/8,
r=(A+B+C)/8+(A+B+C)!8.

Тъй като грубото частно е с недостиг, в някои случаи r ще е по-голям от 6, но няма да надхвърли 7, както може да се провери с изчерпване във формулата. Ако r=7, въпросът се урежда, като r се прави 0 и към q се добавя 1. Алгоритъмът е готов!

Благодарение на възможността много операции да се правят едновременно и за частното, и за остатъка, този алгоритъм се реализира на 6502 по-ефективно от останалите. Разбира се, използвани са и хитрости. Я ми кажете, но без да гледате в решението, до какво може да се съкрати последователността от инструкции AND #7, SEC и SBC #8? Както е казал народът, сиромах човек (асемблерист на „беден" процесор) – жив дявол! След това леко упражнение вече сте готови да разгледате решението.

```
; младшите 8 бита на делимото са в X, а старшият бит – в A
; частното се получава в Y, а остатъкът – в A
; W и H са два работни байта в нулевата страница.
    lsr a
    txa
    ror a
    lsr a
    lsr a
    sta W ; W става делимото/8
    lsr a
    lsr a
    lsr a
    sta H ; H става делимото/64
    clc
    adc W ; след това събиране не може да има пренос!
    tay   ; Y става грубото частно
    lda W
    and #7
    sta W ; W става средната триада на делимото
    txa
    ora #$f8 ; A става младшата триада на делимото – 8
    adc W ; към A се добавя средната триада на делимото
    bcc S1 ; ако няма пренос, текущата сума е по-малка от 8
    iny ; преносът означава и пренос към частното и към
        ; остатъка (флагът C е единица)
    ora #$f8 ; триадата на текущата сума – 8
S1  adc H ; добавя се и старшата триада
    bcc S2 ; като в по-горния случай
    iny
    adc #0
S2  and #7 ; в A е получен грубият остатък
    cmp #7 ; нуждае ли се той от корекция?
    bcc S3 ; не, приключваме
    iny ; да, добавяме към частното 1, а остатъка правим 0
    lda #0
S4  equ *
```