МАШИННА АРИТМЕТИКА В ЧАСОВЕТЕ ПО ИНФОРМАТИКА

Пламен Пенев

Резюме. В статията се прави опит за излизане извън шаблона на преподаване на позиционни бройни системи в училищния курс по информатика. Чрез права и обратна задача действията с двоични числа се пренасят в машинната памет. При наблюдаване на формите на представяне се обръща внимание на някои особености, изясняващи понятието „тип данни“. Въвеждат се елементарни машинни операции, които характеризират работата на изчислителната система.

Keywords: binary numeral system, integer type data, coding of numbers, forward and inverse problem, machine register

1. Увод

Двоичната бройна система е езикът на компютрите на най-ниско ниво. В училищния курс по информатика позиционни бройни системи се изучават в 9. клас. Разглеждат се предимно десетично-двоично и двоично-десетично преобразуване, както и действия с двоични числа. Характерът на материала е такъв, че не се предполага използване на компютър. Поднесени теоретично, чисто математически, тези важни знания остават неинтересни за учениците.

По наше мнение изучаването на двоична бройна система може да се „съжи-ви“, ако действията с двоични числа се пренесат в машинната памет. Разбира се, в този случай се застъпва и темата за представяне на числата в компютъра, но двете теми са взаимосвързани. Тяхното успоредно разглеждане е добра основа за изясняване на едно от основните понятия в информатиката – типа данни. В настоящата статия се изгражда подход, позволяващ редуване на теория и практика още в началните уроци по информатика. Въведените технически подробности, без да изместват същността на материала, способстват за експериментиране и прилагане на компютърни евристики.

2. Помощни средства и техники

2.1. Таблицата на съответствията е удобна за съкратено преобразуване на числата в десетична, двоична и шестнайсетична бройна система - табл. 1.

2.2. За експериментиране в компютърната памет са достатъчни шестнайсетбитовите регистри с общо предназначение Ax, Bx, Dx на процесор i8086 и младшите половинки Al и Bl на първите два. В зависимост от това, дали се работи с цели

Таблица 1.

<10>	<2>	<16>
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

числа, или само с естествени и нула, минималното целочислено представяне се дели на два вида:

– Представяне на числа със знак: диапазон на изменение от –128 до +127 (общо 256 състояния заедно с 0). В този случай първият бит е заделен за знак: 0 означава +, а 1 – минус.
– Представяне на числа без знак: от 0 до 255 ($2^8 – 1$).

2.3. За наблюдение на състоянието и обработка на съдържанието на регистрите използваме вградения в командния интерфейс на Windows системен анализатор Debug. Инструкциите са на машинен мнемокод (с именувани операции).

2.4. Изваждане чрез добавяне на допълнението на умалителя.

Техниката на изваждане чрез събиране е взета от живота. Например в магазина касиерката, допълвайки, връща ресто, което е разлика от изваждането на две числа. Ако извадим 37 от 86, ще получим 49, но разликата може да бъде сметната и другояче: Допълнението на 37 до 100 е 63. Събираме 63 с умаляемото 86 и получаваме 149. Като отнемем кръглото число, до което сме допълнили (100), остава 49.

В двоична бройна система:

$$86_{10} = 56_{16} = 0101\ 0110_2 \text{ (според табл. 1)},$$
$$37_{10} = 25_{16} = 0010\ 0101_2.$$

Ако обозначим с d допълнението на $0010\ 0101_2$ до по–голямото кръгло число, за числата в осемразрядно поле ще получим:

$d_{00100101} = 1\ 0000\ 0000_2 - 0010\ 0101_2 = 1101\ 1011_2$. Това състояние на числото се нарича допълнителен код (ДК). Съществува ефективен машинен алгоритъм за получаване на допълнителния код:

1. Запис на числото в прав код: $1 \mid 010\ 0101_{ПК}$ (първият бит е знаков).
2. Обръщане на всички битове без знаковия (запис в обратен код): $1 \mid 101\ 1010_{ОК}$
3. Добавяне на единица в най-младшия разряд: $1 \mid 101\ 1011_{ДК}$

Операция $0101\ 0110_2 - 0010\ 0101_2$ заместваме с $0101\ 0110_2 + d_{00100101}$.

$$\begin{array}{r} 0101\ 0110_2 \\ +\ 1101\ 1011_2 \\ \hline \end{array}$$

$$10011\ 0001_2 - 1\ 0000\ 0000_2 = 0011\ 0001_2 = 31_{16} = 49_{10}$$

В осемразряден регистър най-старшият бит излиза извън обхвата и се премахва автоматично.

3. Права задача

Задача 1: 27_{10} и -76_{10} се записват съответно в регистри Al и Bl в <2–16> вид (регистрите са осембитови, числата – със знак). Какво ще бъде съдържанието на регистрите след извършване на операция Рг.Al = Рг.Al + Рг.Bl?

$$27_{10} = 1B_{16} = 0001\ 1011_2,$$
$$-76_{10} = -4C_{16} = -0100\ 1100_2.$$

Трасиране на машинното изпълнение:

Таблица 2.

Рг. Al	Рг.Bl	Команди	Коментар	Резултат
$0 \mid 001\ 1011_{пк}$		Mov Al,1B	Запиши $1B_{16}$ в Al	$1B_{16}$
	$1 \mid 100\ 1100_{пк}$	Mov Bl,-4C	Запиши $-4C_{16}$ в Bl	
	$1 \mid 011\ 0011_{ок}$		Вътрешно	
	+ 1		машинен	
	---		алгоритъм	
	$1 \mid 011\ 0100_{дк}$			$B4_{16}$

$+1 \mid 011\ 0100_{дк}$	Add Al,Bl	Събери Al и Bl с резултат в Al
$1 \mid 100\ 1111_{дк}$	<2-16> отрицателна разлика	CF_{16}

Ето въведените инструкции и тяхното изпълнение една по една с помощта на системния анализатор:

```
C:\>debug
-a
0AFC:0100 mov al,1b
0AFC:0102 mov bl,-4c
0AFC:0104 add al,bl
0AFC:0106
-r
      Al          Bl
AX=0000  BX=0000
MOV      AL,1B
-p

AX=001B  BX=0000
MOV      BL,B4
-p

AX=001B  BX=00B4
ADD      AL,BL
-p

AX=00CF  BX=00B4
```

Математическа проверка: $27_{10} - 76_{10} = -49_{10} = -31_{16} = -0011\ 0001_2$

$d_{0011\ 0001} = 1\ 0000\ 0000_2 - 0011\ 0001_2 = 1100\ 1111_2 = CF_{16}$.

Две допълнителни инструкции показват разликата в <2–10> вид.

```
1380:0106 neg al
1380:0108 aam
1380:010A
-p

AX=00CF  BX=00B4
NEG      AL
-p

AX=0031  BX=00B4
AAM
-p

AX=0409  BX=00B4
```

Neg е инструкция за умножение съдържимото на Al по минус 1. В нашия случай действието е равносилно на разкодиране на CF_{16} до прав код с подразбиращ се минус.

Aam преобразува съдържимото на Ax от <2–16> пакетиран вид в <2–10> непакетиран (по една десетична цифра в половинка байт – диапазон за Ax от 0 до 99_{10}).

4. Обратна задача

Ако наречем горната задача права (от въведени числа към крайно състояние на регистрите), то интерес представлява и обратната задача. Можем да я формулираме по следния начин:

Задача 2: След изпълнение на операция Рг.Al=Рг.Al+Рг.Bl съдържимото на Рг.Al е CB_{16}, а на Рг.Bl→ $A2_{16}$. Кои са първоначалните десетични числа, въведени в регистрите?

Числото в Bl ще намерим, като разкодираме допълнителния код до прав и го преобразуваме в <10>.

Рг.BL

$1 \mid 010\ 0010_{ДК}$

$-\qquad 1$

$1 \mid 010\ 0001_{ОК}$

$1 \mid 101\ 1110_{ПК} = -5E_{16} = -94_{10}.$

Al преди операцията ще получим, като от Al след операцията извадим Bl.

Рг.Al

$1 \mid 100\ 1011_{ДК}$

$-1 \mid 010\ 0010_{ДК}$

$0 \mid 010\ 1001_{ПК}$

Първият бит е нула, следователно числото е положително (в прав код) и можем директно да го преобразуваме в <10>: $0010\ 1001_2 = 29_{16} = 41_{10}$. В компютърната памет разсъжденията се проверяват със следните инструкции:

```
C:\>debug
-a
1380:0100 mov al,cb
1380:0102 mov bl,a2
1380:0104 sub al,bl
1380:0106 aam
1380:0108
-r
AX=0000  BX=0000
MOV      AL,CB
-p
```

```
AX=00CB  BX=0000
MOV      BL,A2
-p

AX=00CB  BX=00A2
SUB      AL,BL
-p

AX=0029  BX=00A2
AAM
-p

AX=0401  BX=00A2
```

Математическа проверка:

$$41_{10} - 94_{10} = -53_{10} = -35_{16} = -0011\ 0101_2$$
$$d_{0011\ 0101} = 1\ 0000\ 0000_2 - 0011\ 0101_2 = 1100\ 1011_2 = CB_{16}$$

5. Преобразуване на дробни части

За наблюдение на дробни части в машинната памет е удобна командата IDiv (деление на числа със знак). Делимото се записва в Рг. Dx, делителят в Рг. Bx, а дробната част се търси в Рг.Ax.

Задача 3: Какво ще бъде съдържимото на Рг.Ax, ако $0{,}375_{10}$ се представи в паметта като отношение 3 : 8 с команда IDiv.

Преобразуваме $0{,}375_{10}$ в <16> чрез последователно умножение по 2 и вземане на целите части в прав ред.

$$0{,}375 \times 2 = 0\ |\ ,750$$
$$0{,}750 \times 2 = 1\ |\ ,5$$
$$0{,}5\ \ \times 2 = 1\ |\ ,0$$
$$0{,}0\ \ \times 2 = 0\ |$$

Намираме, че $0{,}375_{10} = 0{,}0110_2 = 0{,}6_{16}$.

Три въведени с Debug инструкции потвърждават получения резултат (двоичната точка се подразбира фиксирана в началото на Рг.Ax):

```
C:\>debug
-a
1380:0100 mov dx,3
1380:0103 mov bx,8
1380:0106 idiv bx
1380:0108
-r
AX=0000  BX=0000  DX=0000
MOV      DX,0003
-p

AX=0000  BX=0000  DX=0003
MOV      BX,0008
-p

AX=0000  BX=0008  DX=0003
IDIV     BX
-p

AX=6000  BX=0008  DX=0000
```

Ако в Dx въведем -3 (FFFD в ДК), след изпълнение на същата команда, в Ax ще получим $A000_{16}$ (допълнителен код на $-0{,}6_{16}$ за 16 разряда).

$$d_{0110\ 0000\ 0000\ 0000} = 1\ 0000\ 0000\ 0000\ 0000$$
$$-\quad 0110\ 0000\ 0000\ 0000$$
$$\text{----------}$$
$$1010\ 0000\ 0000\ 0000$$
$$A\quad 0\quad 0\quad 0_{16}$$

```
C:\>debug
-a
1380:0100 mov dx,-3
1380:0103 mov bx,8
1380:0106 idiv bx
1380:0108
-r
AX=0000  BX=0000  DX=0000
MOV      DX,FFFD
-p
AX=0000  BX=0000  DX=FFFD
MOV      BX,0008
-p
AX=0000  BX=0008  DX=FFFD
IDIV     BX
-p
AX=A000  BX=0008  DX=0000
```

6. Умножение и деление със степени на две

Дописване на нули в края на числото е равносилно на умножение на числото по основата на бройната система, на степен броя на дописаните нули. Това свойство на позиционните бройни системи може да се ползва за бързо умножение и деление.

Задача 4: 47_{10} се въвежда в Рг.Ax в <2–10> вид. Какво ще бъде състоянието на регистъра, ако <2–16> вид на числото се умножи по 2.

$$47_{10} = 2F_{16} = 0010\ 1111_2$$
$$00101111_2$$
$$x\qquad 10_2$$
$$\text{----------}$$
$$00000000$$
$$+\ 0101111_2$$
$$\text{----------}$$
$$01011110_2 = 5E_{16} = 94_{10} = 47x2.$$

В машинната памет:

Таблица 3.

Команди	Коментар	Рг.Ах
Mov Ax,0407	Запис на 47_{10} в <2-10> непакетиран вид	04 07
Aad	Преобразуване на 47_{10} в <2-16> пакетиран вид	00 $2F_{16}$
Shl Ax,1	Умножение съдържимото на Ах по 2 чрез изместване разряд наляво.	00 $5E_{16}$
Aam	Преобразуване на $5E_{16}$ в <2-10> непакетиран вид	09 04

```
C:\>debug
-a
1380:0100 mov ax,0407
1380:0103 aad
1380:0105 shl ax,1
1380:0107 aam
1380:0109
-r
AX=0000  BX=0000
MOV      AX,0407
-p

AX=0407  BX=0000
AAD
-p

AX=002F  BX=0000
SHL      AX,1
-p

AX=005E  BX=0000
AAM
-p

AX=0904  BX=0000
```

Изпълнена 2 пъти, Shl (Shift to left) ще умножи съдържимото на регистъра по 4 и т.н. Чрез допълнително събиране и изваждане на множимото е възможно формиране и на други множители. За бързо деление на 2 се използва командата Shr

(изместване разряд надясно). Естествено точен резултат се получава, ако битът, който се отрязва, е 0.

7. Особености на целочисления тип

При решаване на предложените задачи се изясняват някои въпроси, касаещи целочисления тип данни. Разделянето на интервали зависи от адресируемата единица памет. За числа със знак, когато се адресира един байт (Рг.Al), диапазонът е от -128_{10} до $+127_{10}$ (тип ShortInt в Borland Pascal), за два байта (Рг.Ax) – от –32768 до 32767 (тип Integer), за четири байта – LongInt. Но често учениците бъркат левите и десните граници на диапазоните. Защо например -128_{10} е включено в типа ShortInt ? Отговорът се съдържа в особеностите на кодиране на числа със знак. Имаме случай, при който старшият бит на -80_{16} носи едновременно и знак, и стойност. Ще илюстрираме ситуацията с няколко елементарни инструкции:

```
C:\>debug
-a
1380:0100 mov al,-80
1380:0102 neg al
1380:0104 neg al
1380:0106
-r
AX=0000  BX=0000
MOV      AL,80
-p

AX=0080  BX=0000
NEG      AL
-p

AX=0080  BX=0000
NEG      AL
-p

AX=0080  BX=0000
```

Вижда се, че допълнителният код на $-128_{10} = -80_{16}$ е 80_{16}. След умножение по –1 се получава отново 80_{16} (прав код на $+128_{10}$, но за Ax). Числото със знак се интерпретира по два различни начин, в зависимост от разрядността на полето, в което е представено. Интересно е да се отбележи, че за 80_{16} „претендира“ и –0 в прав код ($1\,|\,000\ 0000_{ПК}$). Но $-0_{дк} = +0_{пк}$. Следователно, състоянието 80_{16} за осембитов регистър е свободно и запазено за -128_{10}. Ако искаме $+128_{10}$ да остане в осем бита, то величината, която го допуска, трябва да е беззнакова (тип Byte в Borland Pascal).

8. Заключение

Описаната методика възстановява баланса между теория и практика при разглеждане на темата за позиционни бройни системи. Обхванати от приложни задачи, двоично-десетичните преобразования придобиват конкретен смисъл. Писането и тестването на програми директно в машинната памет позволяват да се направят допълнителни изводи, хвърлящи светлина върху математическите основи на компютрите:

1. Шестнайсетичната бройна система е посредник между „десетичния човек“ и „двоичната машина“. Тя ускорява четенето на машинната памет.

2. Обратният код е преходно състояние от прав към допълнителен код на отрицателните числа.

3. Допълнителният код е удобен за съхраняване на отрицателни числа.

4. Машинната аритметика е максимално опростена. Операцията „изваждане“ се замества с операции „инверсия“ (обръщане) и „събиране“.

И не на последно място упражненията са добра предварителна подготовка за изучаване на програмен език от по-високо ниво.

ЛИТЕРАТУРА

Азълов, П. (2005). Еквивалентни променливи. *Математика и информатика*, 1, 3 – 12.

Скэнлон Л. (1989). *Персональные ЭВМ IBM PC и XT. Программирование на языке ассемблера*. Москва: Радио и связь.

MACHINE ARITHMETIC IN INFORMATICS CLASSES

Abstract. An attempt is done in the paper to break out of the teaching pattern concerning positional numeral systems in the Informatics curriculum. Operations with binary numbers are transferred to the machine memory by forward and inverse problems. Some peculiarities clarifying the “data type” notion are emphasized in the observation process of presenting forms. Elementary machine operations are introduced, thus characterizing the computing system performance.

Plamen Penev
✉ Teacher
„Panayot Volov“ school
100, Soedinenie Street
9700 Shumen, Bulgaria
E-mail: plampenev@abv.bg