Двоично - десятичная арифметика

Поскольку человеку наиболее привычны представление и арифметика в десятичной системе счисления, а для компьютера - двоичное представление и двоичная арифметика, была введена компромиссная система ***двоично-десятичной записи*** чисел.

Такая система чаще всего применяется:

- там, где существует необходимость частого использования процедуры **десятичного ввода-вывода** (электронные часы, калькуляторы, АОНы и т.п., ввод привычных человеку цифр с клавиатур),
- **финансовые вычисления**,
- для **хранения и обработки чисел с большим количеством цифр**.

Организация вычислений на компьютере решает 3 основные задачи:

- ВВОД ДАННЫХ,
- ОБРАБОТКУ ДАННЫХ
- ВЫВОД ДАННЫХ.

В простейшем случае ВВОД ДАННЫХ осуществляется пользователем, а ОБРАБОТКА ДАННЫХ и ВЫВОД ДАННЫХ - компьютером.

На всех этих этапах происходит **преобразование данных:**

- пользователю удобно ***вводить*** числа в **десятичной системе счисления** (ВВОД ДАННЫХ),
- компьютер вынужден ***преобразовывать*** их в **двоичные числа** и в них же ***производить вычисления*** на стадии ОБРАБОТКИ ДАННЫХ
- и снова ***преобразовывать полученные результаты*** в **десятичную систему счисления**, для ВЫВОДА ДАННЫХ в понятном пользователю виде.

Двоично-десятичный код (***binary-coded decimal***), **BCD**, 8421-BCD — форма записи чисел, когда каждая десятичная цифра числа записывается в виде его четырёхбитного двоичного кода.

При помощи четырёх бит можно закодировать шестнадцать цифр. Из них используются 10. Остальные 6 комбинаций в двоично-десятичном коде являются запрещенными.

Двоично-десятичный код				Десятичный код
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

Микропроцессоры используют «**чистые**» **двоичные числа**, однако **имеют команды преобразования в двоично-десятичную запись**.

Полученные двоично-десятичные числа легко представимы в десятичной записи, более понятной людям.

Десятичное число	3	6	9	1
Двоично-десятичное число	0011	0110	1001	0001
«Чистое» двоичное число	0000	1110	0110	1011

Достоинства

- **Упрощён вывод чисел** на индикацию — вместо последовательного деления на 10 требуется просто вывести на индикацию каждый полубайт. Аналогично, проще ввод данных с цифровой клавиатуры.
- Для дробных чисел (как с фиксированной, так и с плавающей точкой) при переводе в человекочитаемый десятичный формат и наоборот **не теряется точность**.
- **Упрощены умножение и деление на 10**, а также округление.

Недостатки

- Требует **больше памяти**.
- **Усложнены арифметические операции**.

Так как в 8421-BCD используются только 10 возможных комбинаций 4-х битового поля вместо 16, существуют запрещённые комбинации битов: $1010(10_{10})$, $1011(11_{10})$, $1100(12_{10})$, $1101(13_{10})$, $1110(14_{10})$ и $1111(15_{10})$.

Поэтому, при сложении и вычитании чисел формата 8421-BCD действуют следующие **правила**:

Правила

- При **сложении** двоично-десятичных чисел каждый раз, когда происходит перенос бита в старший полубайт, необходимо к полубайту, от которого произошёл перенос, добавить корректирующее значение 0110_2
(= $6_{10} = 16_{10} — 10_{10}$: разница количеств комбинаций полубайта и используемых значений).

Правила

- При **сложении** двоично-десятичных чисел каждый раз, когда встречается недопустимая для полубайта комбинация (число, большее 9), необходимо к каждой недопустимой комбинации добавить корректирующее значение 0110_2 с **разрешением переноса (!)** в старшие полубайты.
- При **вычитании** двоично-десятичных чисел, для каждого полубайта, получившего заём из старшего полубайта, необходимо провести коррекцию, отняв значение 0110_2.

Правила

Пример:

Требуется: Найти A = D + C, где D = 3927, C = 4856

Решение:

Представим числа D и C в двоично-десятичной форме:

$D = 3927_{10} = 0011\ 1001\ 0010\ 0111_{BCD}$

$C = 4856_{10} = 0100\ 1000\ 0101\ 0110_{BCD}$

Правила

Суммируем числа D и C по правилам двоичной арифметики:

```
        *           **
  0011 1001 0010 0111
+ 0100 1000 0101 0110
  ____________________
= 1000 0001 0111 1101 - Двоичная сумма
+      0110      0110 - Коррекция
  ____________________
  1000 0111 1000 0011
```

'*' — тетрада, из которой был перенос в старшую тетраду

'**' — тетрада с запрещённой комбинацией битов

В тетраду, помеченную символом *, добавляем шестёрку и **игнорируем** перенос (если бы он был) в старшую тетраду;

В тетраду, помеченную символом **, добавляем шестёрку и (!) **разрешаем** распространение переноса.

- A = 8783

- Эти «Правила» можно реализовать **программно** (т.е. реализуем алгоритмически сами)
- Можно реализовать **в системе команд** процессора и использовать их («Правила» уже реализованы соответствующими командами МП)

* В лабах сделать и так, и так!

Упакованные BCD числа

- Каждый байт хранит 2 десятичных цифры – в младшем и старшем полубайте (количество байт = количество цифр/2)

- Пример: 23_{10}
- упакованное BCD: 0x23 = 0010 0011
- шестнадцатеричное: 0x17 = 0001 0111

Упакованные BCD числа, арифметика

- Обычные арифметические операции над BCD дают **неправильный результат**

- Пример:

десятичное Сложение	сложение BCD как двоичных чисел	должно быть как BCD
23+18=42	0x23+0x18 = 0x3b	0x42

- Нужна **коррекция**!

BCD-коррекция упакованных чисел

команда	**после**	данные
daa	сложения	AL
das	вычитания	

- Вызываются сразу после арифметического действия (МП обрабатывает числа в «чистом» двоичном формате!) и корректируют результат
- Флаги:
  - **AF** – если был перенос из 1-й цифры
  - CF – если был перенос из 2-й цифры
  - устанавливают SF, ZF, PF

Пример a := 1234 + 5678

```
a  db 12h,34h,  ; возможно переполнение (см. CF)
b  db 56h,78h,

mov al, a+1   ; al = 0x34
add al, b+1   ; al = 0xAC, CF = 0
daa           ; al = 0x12, CF = 1
mov a+1, al

mov al, a       ; al = 0x12
adc al, b       ; al = 0x69, CF = 0
daa             ; al = 0x69, CF = 0
mov a, al
```

Неупакованные BCD числа

- Каждый байт хранит одну десятичную цифру (сколько цифр – столько и байт)

- Пример: 23_{10}
- неупакованное BCD: 0x02 0x03
- упакованное BCD: 0x23

BCD-коррекция неупакованных чисел

команда	**после**	источник	приемник
aaa	сложения	AL	AX
aas	вычитания		

- Если есть перенос из единственной **цифры**, то выполняется

  AH := AH +1 для aaa

  AH := AH - 1 для aas

  OF := 1

  CF := 1

Пример a := 0234 + 0678

```
a  db 0, 2, 3, 4
b  db 0, 6, 7, 8

mov al, a+3
mov ah, a+2      ; ah=3, al=4
add al, b+3      ; ah=3, al=12, CF=0
aaa              ; ah=4, al=2,  CF=0
mov a+3, al

mov al, ah
mov ah, a+1      ; ah=2, al=4
add al, b+2      ; не adc, перенос уже в al !
aaa
..
```

BCD-коррекция неупакованных чисел

команда	источник	приемник	**после**
aam	al	ax	**умножения**

- Произведение двух цифр занимает 2 цифры
- Выполняет:

  AH := AL / 10 (старшая цифра)

  AL := AL % 10 (младшая цифра)

  устанавливает SF, ZF, PF

Пример a := 34 * 6

```
a  db 0, 3, 4      ; 0 – для переноса
b  db 6

mov al, a+2   ; al = 4
mul b         ; ah=0, al=24
aam           ; ah=2, al=4
mov a+2, al

mov al, a+1   ; al = 3
mul b         ; al = 18
add al, ah    ; al = 20 (добавлен перенос)
aam           ; ah=2, al=0
mov a+1, al
...
```

BCD-коррекция неупакованных чисел

- Как видно из примера, **aam** позволяет выполнять умножение длинных чисел без использования **aaa !**

BCD-коррекция неупакованных чисел

команда	источник	приемник	**до**
aad	ax	ax	**деления**

- Подготавливает делимое:

  AX = 10*AH + AL

- Флаги – как у **aam**

Пример a := 567 / 3

```
a  db 5, 6, 7
b  db 3

xor ah, ah

mov al, a+0    ; ah=0, al=5
aad            ; ah=0, al=5
div b          ; ah=2, al=1
mov a+0, al

mov al, a+1    ; ah=2, al=6
aad            ; ah=0, al=26
div b          ; ah=2, al=8
mov a+1, al
...
```

```
 567 | 3
-3   | 189
 26
-24
  27
 -27
   0
```

BCD-коррекция неупакованных чисел

- В предыдущем примере можно было использовать для подготовки первого делимого

```
movsx ax, a
```