Представление целых чисел в ограниченной двоичной разрядной сетке (РС) компьютера

Для хранения целой переменной в памяти компьютера используется фиксированное заранее известное число бит. Например, для хранения *a*=2 в компьютерную память будет записано следующее двоичное число, если используется 32-разрядный компьютер:

$$00000000000000000000000000000010_{(2)}.$$

Процессор за один такт работы выполняет операцию сразу со всеми 32-мя битами:

$$\begin{array}{r} {}_{+}00000000000000000000000000000010_{(2)} \\ \underline{00000000000000000100000000000010}_{(2)} \\ 00000000000000000100000000000100_{(2)} \end{array}$$

Пусть для хранения целого неотрицательного числа в переменной *a* используется *k* бит.

$$\mathrm{MIN}(a) = 000\ldots000_{(2)} = 0,$$

$$\mathrm{MAX}(a) = 111\ldots111_{(2)} = 2^k - 1.$$

$$\boxed{\begin{array}{l} 999 = 1000 - 1 = 10^3 - 1 \\ 111_{(2)} = 1000_{(2)} - 1 = 2^3 - 1 \end{array}}$$

Диапазон представления целых неотрицательных чисел в k-разрядной сетке: **от 0 до 2^k-1.**

Представление целых чисел со знаком в компьютере

В ЭВМ нет способа обозначить в двоичной СС знак «МИНУС» перед числом. Способы решения этой проблемы с примерами для 4-разрядного компьютера:

- **Специальный знаковый бит (СЗБ)**
  $+5 = 0101_2$, $-5 = 1101_2$ (первый бит означает знак числа)
- **Фиксированное смещение влево (ФСВ)**
  $-5 = 0000_2$, $-4 = 0001_2$, …, $+10 = 1111_2$ (все числа уменьшены на 5)
- **Нега-двоичная система счисления (НДСС)**
  $-5 = 1111_{-2}$, $+5 = 0101_{-2}$ (основание СС равно «-2»).
- **Обратный/инверсный код (ОК)**
  $+5 = 0101_2$, $-5 = 1010_2$ (инвертируются все биты)
- **Дополнительный код (ДК)**
  $+5 = 0101_2$, $-5 = 1011_2$ (инвертировать все биты и прибавить 1)

Целые числа со знаком в трёхразрядном компьютере

Для сравнения – диапазон представления целых неотрицательных чисел в трёхразрядной сетке: от $000_{(2)}$ до $111_{(2)}$, т. е. от 0 до 7.

Трёхразрядный код	СЗБ	ФСВ (5)	НДСС	ОК	ДК
000	+0	-5	0	+0	0
001	1	-4	1	1	1
010	2	-3	-2	2	2
011	3	-2	-1	3	3
100	-0	-1	4	-3	-4
101	-1	0	5	-2	-3
110	-2	1	2	-1	-2
111	-3	2	3	-0	-1
Диапазон	-3..+3	-5..+2	-2..5	-3..+3	-4..+3

Целые числа со знаком в n-разрядном компьютере

Имея n-разрядный двоичный регистр, можно закодировать 2^n разных символов. Для кодирования целых чисел без знака используется диапазон от 0 до $2^n - 1$. Каков диапазон хранимых чисел со знаком в n-разрядном регистре?

1. Специальный знаковый бит (СЗБ):
   от $-(2^{n-1} - 1)$ до $+(2^{n-1} - 1)$.

min→	1	1	1	1	...	1	1	1	1
max→	0	1	1	1	...	1	1	1	1

2. Фиксированное смещение влево (ФСВ):
   от $(-S)$ до $(2^n - 1 - S)$, где S – смещение.

0	0	0	0	...	0	0	0	0
1	1	1	1	...	1	1	1	1

3. Нега-двоичная система счисления (НДСС):
   чётное n: от $-(2^n-1)*2/3$ до $(2^n-1)/3$
   нечётное n: от $-(2^{n-1}-1)*2/3$ до $(2^{n+1}-1)/3$
   любое n: от $-(2^{n-(n \bmod 2)}-1)*2/3$ до $(2^{n+(n \bmod 2)}-1)/3$

...	1	0	1	0	1	0	1	0
...	0	1	0	1	0	1	0	1

4. Обратный/инверсный код (ОК):
   от $-(2^{n-1} - 1)$ до $+(2^{n-1} - 1)$.

1	0	0	0	...	0	0	0	0
0	1	1	1	...	1	1	1	1

5. Дополнительный код (ДК):
   от (-2^{n-1}) до $(2^{n-1}-1)$.

1	0	0	0	...	0	0	0	0
0	1	1	1	...	1	1	1	1

Дополнительный код: пример

Как хранится число «-2» в памяти десятиразрядного компьютера?

Решение

1 шаг: записать число «+2», используя все доступные разряды

$2_{10} = 0000000010_2$

2 шаг: инвертировать каждый бит полученного числа:

$0000000010_2 \rightarrow 1111111101_2$

3 шаг: прибавить один

$$\begin{array}{r} 1111111101_2 \\ + \underline{0000000001_2} \\ 1111111110_2 \end{array}$$

4 шаг: радоваться результату: $-2_{10} = 1111111110_2$ (обратный перевод выполняется так же)

Иллюстрация эффекта $2 + (-2) = 0$ $\rightarrow$

$$\begin{array}{r} {}_{+}0000000010_2 \\ \underline{1111111110_2} \\ 10000000000_2 \end{array}$$ – это ноль, т. к. 11-го разряда нет

Почему дополнительный код

$$
\begin{array}{r}
+\;\; 0\,1\,1\,1_2 \\
\underline{1\,0\,1\,1_2} \\
\mathbf{1}\,0\,0\,1\,0_2
\end{array}
$$

СЗБ	НДСС	ОК	ДК
$+7$	$+3$	$+7$	$+7$
$+\;\underline{-3}$	$+\;\underline{-9}$	$+\;\underline{-4}$	$+\;\underline{-5}$
$+2$	-2	$+2$	$+2$

Как придумали правило ДК? Почему нужно инвертировать биты и прибавлять 1?

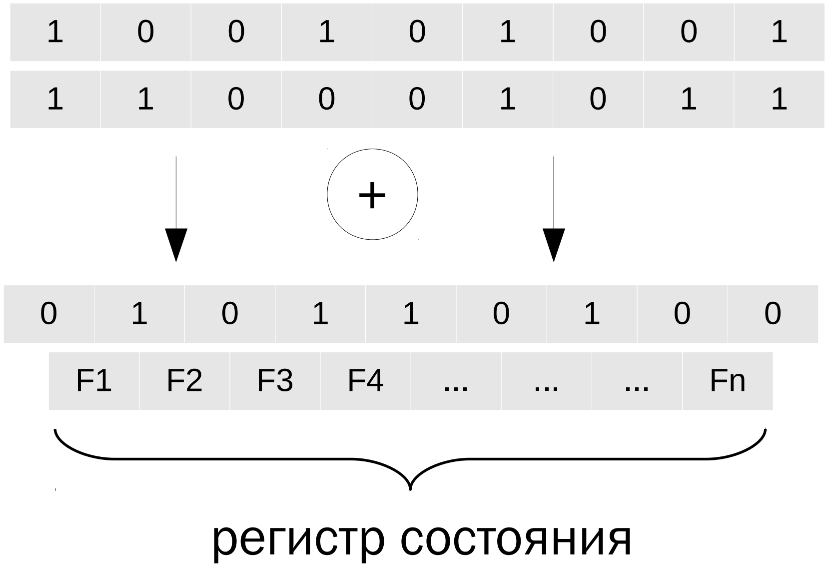
$x_{(2,n)} + \mathrm{inv}(x_{(2,n)}) = \ldots 11111111_{(2,n)} = 2^n - 1.$ Пример: $1111_{(2,4)} = 2^4 - 1$

$\mathrm{inv}(x_{(2,n)}) + 1 = 2^n - x_{(2,n)}$

$\mathrm{inv}(x_{(2,n)}) + 1 = -x_{(2,n)}$

$a_{(2,n)} - b_{(2,n)} = a_{(2,n)} + (-b_{(2,n)}) = a_{(2,n)} + (2^n - b_{(2,n)}) = a_{(2,n)} + (\mathrm{inv}(b_{(2,n)}) + 1)$

- После любой арифметической операции процессор автоматически без явной команды от программиста устанавливает флаги, характеризующие состояние процессора.
- Совокупность этих флагов называется регистром состояния.
- Программист может анализировать содержимое регистра состояния процессора для принятия решений в программе.



if (F1 == 0) then ... else ...;

Флаги состояния процессора

***SF – Sign Flag*.** Равен 1, если результат операции отрицателен, иначе – 0.

***ZF – Zero Flag*.** Равен 1, если результат операции равен нулю.

***PF – Parity Flag*.** Равен 1, если младший байт результата выполнения операции содержит чётное число единиц.

***AF – Adjust Flag*.** Равен 1, если произошёл заём или перенос между первым и вторым полубайтом (нибблом).

***CF – Carry Flag*.** Равен 1, если происходит перенос за пределы разрядной сетки или заём извне.

OF – Overflow Flag. Равен 1, если результат операции не помещается разрядную сетку (при использовании дополнительного кода).

Флаги переполнения и переноса (OF, CF)

OF – Overflow Flag. Принимает значение 1, если в результате выполнения операции со знаковыми числами появляется одна из ошибок:

1) складываем положительные числа, получаем отрицательный результат;
2) складываем отрицательные числа, получаем положительный результат.

Примеры для 4-разрядного компьютера:

$0100_{(2)} + 0001_{(2)} = 0101_{(2)}$ (CF=0, OF=**0**) : +4 + 1 = 5

$0110_{(2)} + 1001_{(2)} = 1111_{(2)}$ (CF=0, OF=**0**) : +6 - 7 = -1 (1111_2 в доп. коде это -1_{10})

$1000_{(2)} + 0001_{(2)} = 1001_{(2)}$ (CF=0, OF=**0**) : -8 + 1 = -7

$1100_{(2)} + 1100_{(2)} = 1000_{(2)}$ (CF=1, OF=**0**) : -4 - 4 = -8

$1000_{(2)} + 1000_{(2)} = 0000_{(2)}$ (CF=1, OF=**1**) : -8 - 8 = 0

$0101_{(2)} + 0100_{(2)} = 1001_{(2)}$ (CF=0, OF=**1**) : +5 + 4 = -7

Пример установки флагов состояния процессора

16-разрядный компьютер

Пример 1

$$
\begin{array}{rrr}
 & 0010.0101.0000.1100_{(2)} & +9484_{(10)} \\
+ & \underline{0011.1101.1010.0100}_{(2)} & +15780_{(10)} \\
 & 0110.0010.1011.0000_{(2)} & = +25264_{(10)}
\end{array}
$$

$\boxed{CF=0,\ OF=0,\ ZF=0,\ AF=1,\ SF=0,\ PF=0}$

Пример 2

$$
\begin{array}{rrr}
 & 0110.0010.1010.1001_{(2)} & +25257_{(10)} \\
+ & \underline{0011.1101.1010.1100}_{(2)} & +15788_{(10)} \\
 & 1010.0000.0101.0101_{(2)} & = -24491_{(10)}
\end{array}
$$

$\boxed{CF=0,\ OF=1,\ ZF=0,\ AF=1,\ SF=1,\ PF=1}$

Пример 3

$$
\begin{array}{rrr}
 & 1110.0111.0110.1000_{(2)} & -6296_{(10)} \\
+ & \underline{0110.0010.1011.0000}_{(2)} & +25264_{(10)} \\
 & 1.0100.1010.0001.1000_{(2)} & = +18968_{(10)}
\end{array}
$$

$\boxed{CF=1,\ OF=0,\ ZF=0,\ AF=0,\ SF=0,\ PF=1}$