Представление целых чисел в памяти ЭВМ

Для представления чисел в ЭВМ обычно используют битовые наборы — последовательности нулей и единиц фиксированной длины. Организовать обработку наборов фиксированной длины технически легче, чем наборов переменной длины. Позиция в битовом наборе называется разрядом. В ЭВМ разрядом называют также часть регистра (или ячейки памяти), хранящую один бит.

Для представления знаковых целых чисел используются четыре способа:

1. прямой код;
2. код со сдвигом;
3. обратный код;
4. дополнительный код.

Прямой код

При записи числа в прямом коде старший разряд является знаковым разрядом. Если его значение равно нулю, то представлено положительное число или положительный ноль, если единице, то представлено отрицательное число или отрицательный ноль. В остальных разрядах (которые называются цифровыми) записывается двоичное представление модуля числа. Например, число −10 в восьмибитном типе данных, использующем прямой код, будет выглядеть так: 10001010. В шестнадцатибитном типе данных: 1000000000001010. Число 10 в восьмибитном типе данных, использующем прямой код, будет выглядеть так: 00001010. В шестнадцатибитном типе данных: 0000000000001010.

Таким способом в n-битовом типе данных можно представить диапазон чисел $[-2^{n-1}+1;\ 2^{n-1}-1]$.

Достоинства представления чисел с помощью прямого кода

1. Получить прямой код числа достаточно просто.
2. Из-за того, что 0 обозначает +, коды положительных чисел относительно беззнакового кодирования остаются неизменными.
3. Количество положительных чисел равно количеству отрицательных.

Недостатки представления чисел с помощью прямого кода

1. Выполнение арифметических операций с отрицательными числами требует усложнения архитектуры центрального процессора (например, для вычитания невозможно использовать сумматор, необходима отдельная схема для этого).
2. Существуют два нуля: +0 (000…000) и −0 (100…000), из-за чего усложняется арифметическое сравнение.

Код со сдвигом

При использовании кода со сдвигом минимальное отрицательное число, которое может храниться в типе данных, кодируется последовательностью нулей. Число на единицу больше кодируется единицей и т.д.

Алгоритм кодирования:

1. к кодируемому числу прибавляют $\mathbf{2^{n-1}}$;
2. переводят получившееся число в двоичную систему исчисления.

Можно получить диапазон значений $[-\mathbf{2^{n-1}}; \mathbf{2^{n-1}-1}]$.

Например, число −10 в восьмибитном типе данных, будет выглядеть так: $-10 + 2^{8-1} = -10 + 128 = 118 = 01110110$. В шестнадцатибитном типе данных: $-10 + 2^{16-1} = -10 + 32768 = 32758 = 0111111111110110$. Число 10 в восьмибитном типе данных, будет выглядеть так: $.10 + 2^{8-1} = 10 + 128 = 138 = 10001010$. В шестнадцатибитном типе данных: $10 + 2^{16-1} = 10 + 32768 = 32778 = 1000000000001010$.

Достоинства представления чисел с помощью кода со сдвигом

1. Не требуется усложнение архитектуры процессора.
2. Нет проблемы двух нулей.

Недостатки представления чисел с помощью кода со сдвигом

1. При арифметических операциях нужно учитывать смещение, то есть проделывать на одно действие больше (например, после «обычного» сложения двух чисел у результата будет двойное смещение, одно из которых необходимо вычесть).
2. Ряд положительных и отрицательных чисел несимметричен.

Из-за необходимости усложнять арифметические операции код со сдвигом для представления целых чисел используется не часто, но зато применяется для хранения порядка вещественного числа.

Обратный код

Алгоритм получения кода числа:

- если число положительное, то в старший разряд (который является знаковым) записывается ноль, а далее записывается само число;
- если число отрицательное, то код получается инвертированием представления модуля числа (получается обратный код);
- если число является нулем, то его можно представить двумя способами: +0 (000…000) или −0 (111…111).

Можно получить диапазон значений $[-\mathbf{2^{n-1}+1}; \mathbf{2^{n-1}-1}]$.

Например, число −10 в восьмибитном типе данных, будет выглядеть так: 00001010 → 11110101. В шестнадцатибитном типе данных: 0000000000001010 → 1111111111110101. Число 10 в восьмибитном типе данных, будет выглядеть так: .00001010. В шестнадцатибитном типе данных: 0000000000001010.

Достоинства представления чисел с помощью кода с дополнением до единицы

1. Простое получение кода отрицательных чисел.
2. Из-за того, что 0 обозначает +, коды положительных чисел относительно беззнакового кодирования остаются неизменными.
3. Количество положительных чисел равно количеству отрицательных.

Недостатки представления чисел с помощью кода с дополнением до единицы

1. Выполнение арифметических операций с отрицательными числами требует усложнения архитектуры центрального процессора.
2. Существуют два нуля: +0 и −0.

Дополнительный код

Алгоритм получения кода числа:

- если число положительное, то в старший разряд (который является знаковым) записывается ноль, а далее записывается само число;
- если число отрицательное, то к обратному коду числа прибавляется 1;

Можно получить диапазон значений $[-\mathbf{2}^{\boldsymbol{n-1}}; \mathbf{2}^{\boldsymbol{n-1}} - \mathbf{1}]$.

Например, число −10 в восьмибитном типе данных, будет выглядеть так: $00001010 \rightarrow 11110101 + 1 = 11110110$. В шестнадцатибитном типе данных: $0000000000001010 \rightarrow 1111111111110101 + 1 = 1111111111110110$. Число 10 в восьмибитном типе данных, будет выглядеть так: $.00001010$. В шестнадцатибитном типе данных: 0000000000001010.

Достоинства представления чисел с помощью кода с дополнением до двух

1. Возможность заменить арифметическую операцию вычитания операцией сложения и сделать операции сложения одинаковыми для знаковых и беззнаковых типов данных, что существенно упрощает архитектуру процессора и увеличивает его быстродействие.
2. Нет проблемы двух нулей.

Недостатки представления чисел с помощью кода с дополнением до двух

1. Ряд положительных и отрицательных чисел несимметричен, но это не так важно: с помощью дополнительного кода выполнены гораздо более важные вещи, желаемые от способа представления целых чисел.
2. В отличие от сложения, числа в дополнительном коде нельзя сравнивать как беззнаковые, или вычитать без расширения разрядности.

Несмотря на недостатки, дополнение до двух в современных вычислительных системах используется чаще всего.