Организация ЭВМ и вычислительных систем

Практические занятия №1–3. Системы счисления, кодирование и арифметика целых чисел

Макаров П. А.

Содержание

Цель 1

1 Краткая теория 1
1.1 Системы счисления 1
1.2 Целочисленные числовые коды с учётом знака 2
1.3 Альтернативные системы кодирования 3

2 Практические задания и контрольные вопросы 5

Источники 6

Цель

- Освоение основных систем счисления, применяемых в вычислительной технике.
- Изучение различных кодов, используемых для представления целых чисел.
- Приобретение навыков выполнения арифметических операций над целыми числами.

1 Краткая теория

1.1 Системы счисления

В вычислительной технике используются следующие системы счисления.

- **Двоичная** (основание $B = 2$), значащие цифры: `0`, `1`.

- **Восьмеричная** (основание $B = 8$), цифры: `0`–`7`.
- **Шестнадцатеричная** (основание $B = 16$), цифры: `0`–`9`, `A`–`F`.

Пример перевода: $\texttt{157}_{10} = \texttt{10011101}_2 = \texttt{235}_8 = \texttt{9D}_{16}$.

1.2 Целочисленные числовые коды с учётом знака

Двоичная система счисления позволяет "напрямую" (т. е. прямым отождествлением) кодировать только **положительные целые числа и ноль** $\mathbb{N}_0$. Выход за рамки множества $\mathbb{N}_0$ требует более сложной системы кодирования[1].

Так, для представления отрицательных чисел сравнительно часто на практике используются следующие кодовые системы.

1. **Прямой код:** старший бит задаёт знак числа (обычно для положительных чисел используется `0`, для отрицательных — `1`).
2. **Обратный код:** индикация знака как в прямом коде и инверсия всех значащих битов числа.
3. **Дополнительный код:** обратный код + 1.
4. **Системы со смещением:** исходное число суммируется с заданным смещением (как правило, 2^{m-1} или $2^{m-1} - 1$).

Исходное число	**Коды:**			
	прямой	**обратный**	**дополнительный**	**excess-128**
-5	10000101	11111010	11111011	01111011
-10	10001010	11110101	11110110	01110110

Таблица 1: Примеры записи отрицательных чисел с помощью $m = 8$ битов

В связи с этой таблицей очень полезно изучить работу программы, исходный код которой на языке C приведён ниже.

```c
#include <stdio.h>

int main() {
  int x = -5;
  int i = 128;
  printf("The number %d is encoded as ", x);
  while(i > 0) {
```

[1] Заметим, что такая система далеко не единственна. Более того, число различных кодировок с точки зрения математики формально (до тех пор, пока не уточняются конкретные дополнительные требования) ничем не ограничено!

```
8       printf("%d", (x & i) ? 1 : 0);
9       i = i >> 1;
10    }
11    printf(".\n");
12    printf("And more \"honest\" code is 0x%X.\n", x);
13    return 0;
14  }
```

1.3 Альтернативные системы кодирования

Отметим, что представление даже неотрицательных целых чисел $\mathbb{N}_0$ в вычислительной технике не ограничивается только "обычным двоичным кодом". Существуют и другие кодировки, к которым, в частности, относится так называемый двоично-десятичный код (BCD — Binary-Coded Decimal).

Строго говоря, BCD — это общее название группы так называемых **взвешенных кодов**[2]. Код называется взвешенным, потому что каждому двоичному разряду приписывается некоторый "вес". При этом значение тетрады — группы из четырёх битов, кодирующей данную десятичную цифру, определяется суммой весов тех битов, в которых присутствуют 1. Более конкретно, под BCD чаще всего имеют в виду именно код 8421-BCD, согласно которому каждый десятичный разряд числа представляется тетрадой, содержащей обычный двоичный код данного разряда.

Пример: десятичное число 217_{10} в двоично-десятичном коде 8421-BCD будет записано как $0010\ 0001\ 0111_{\text{BCD}}$.

Вообще говоря, с помощью четырёх бит можно закодировать 16 цифр, однако в системах BCD из них используются только десять. В связи с этим остальные шесть комбинаций являются запрещёнными. Из этого факта вытекают как достоинства, так и недостатки двоично-десятичных кодов.

Преимуществами BCD являются следующие особенности.

1. Упрощён вывод чисел на индикацию — вместо последовательного деления двоичного кода на 10 требуется просто вывести на семисегментный индикатор каждый полубайт — ниббл. Аналогично, проще оказывается и ввод данных с цифровой клавиатуры.

2. Для дробных чисел (как с фиксированной, так и с плавающей запятой) при переводе в человекочитаемый десятичный формат и наоборот не теряется точность.

3. Упрощены умножение и деление на 10, а также округление.

По этим причинам двоично-десятичный формат применяется как в бытовых калькуляторах, так и более универсальных компьютерах.

[2] Более того, существуют и невзвешенные коды (например, Excess-3), в которых значение цифры нельзя получить простым суммированием фиксированных весов разрядов.

Недостатки двоично-десятичных кодов:

1. требуют больше памяти;
2. усложнены арифметические операции.

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

1. При сложении двоично-десятичных чисел каждый раз, когда происходит перенос бита в старший полубайт, необходимо к полубайту, от которого произошёл перенос, добавить корректирующее значение $\texttt{0110}_2$.
2. При сложении двоично-десятичных чисел каждый раз, когда встречается недопустимая для полубайта комбинация (число, большее 9), необходимо к каждой недопустимой комбинации добавить корректирующее значение $\texttt{0110}_2$ с разрешением переноса в старшие полубайты.
3. При вычитании двоично-десятичных чисел, для каждого полубайта, получившего заём из старшего полубайта, необходимо провести коррекцию, отняв значение $\texttt{0110}_2$.

В связи со всем сказанным в данном разделе по поводу BCD, очень полезно изучить в отладчике DEBUG процесс исполнения программы для процессоров семейства x86, фрагмент исходного текста которой на языке ассемблера в синтаксисе Intel приведён ниже.

```
1 mov ax, 1726
2 mov bx, 3438
3 add ax, bx
4 daa
```

Помимо 8421-BCD существуют и другие взвешенные двоично-десятичные коды, например: 2421 (код Айкена), 4221, 5421, коды с отрицательными весами, такие как $84\bar{2}\bar{1}$ и т. п. Эти коды также имеют свои преимущества[3] и недостатки, но в настоящее время крайне редко используются.

Более того, помимо “классического” BCD существуют различные варианты так называемых «уплотнённых десятичных чисел». Например, каждые три десятичные цифры можно кодировать с помощью десяти битов[4]. В этом примере важно то, что данное кодирование оказывается эффективным из-за преобразования между 10-битным кодом и тремя отдельными десятичными цифрами, которое можно осуществить с помощью простой и быстрой логической схемы. Именно поэтому такая система кодирования используется в десятичных числах с плавающей запятой, описанных в стандарте IEEE 754-2008.

[3] В частности, код Айкена является самодополняющим, что может быть удобно для некоторых арифметических операций.

[4] Почему это так? В чём здесь выигрыш по сравнению с обычными BCD-кодами?

В завершении отметим, что существуют и альтернативные для BCD системы кодирования, в частности — кодирование «2 из 5», десятичные коды Грея, би-квинарные коды и другие специальные представления, не являющиеся взвешенными кодами. Код «2 из 5» использовался, например, в компьютерах серий IBM 7070, IBM 7072 и IBM 7074.

2 Практические задания и контрольные вопросы

1. Что такое система счисления и какие из них вам известны? Что общего между всеми системами, перечисленными в разделе 1.1?

2. Переведите числа 111_{10}, 125_{10}, 255_{10} и 1024_{10} в двоичную, восьмеричную и шестнадцатеричную системы. Почему в современной вычислительной технике используется именно двоичная система счисления?

3. Выполните обратный перевод для чисел $\texttt{1101011}_2$, $\texttt{3F7}_{16}$ и $\texttt{755}_8$.

4. Выразите числа `3Ah`, `271o`, `99`, `111101b` в основных системах счисления. В связи с этой задачей подумайте над следующими вопросами. Почему во многих языках программирования запрещено давать идентификаторам (именам переменных, функций и т. п.) имена, начинающиеся с символа цифры? Полностью ли решает этот запрет имеющуюся проблему? Зачем была придумана префиксная форма записи чисел (ведущие `0x`, `0` и `0b`)?

5. Запишите числа -24_{10}, -42_{10}, -101_{10} и -128_{10} двоичным кодом в четырёх системах представления отрицательных чисел. Используйте 8 разрядов.

6. Проанализируйте работу программы, исходный код которой приведён в конце раздела 1.2. Замените в этом листинге четвёртую, а также шестую строки на следующие:

```
4 unsigned x = -5;
```

   и

```
6 printf("The number %u is encoded as ", x);
```

   Почему транслятор не отреагирует на “ошибку” в четвёртой строке, допуская инициализацию переменной `x` значением `-5`, хотя теперь её тип явным образом указан нами как беззнаковый? Что произойдёт, если изменить только четвёртую строку, не меняя шестую?

7. Выполните сложение в дополнительном коде:

   - $25 + (-37)$;
   - $(-50) + (-30)$;
   - $100 + (-100)$.

В чем преимущество дополнительного кода перед прямым? Как обнаружить переполнение при сложении в дополнительном коде?

8. В чём разница между *двоичной системой счисления* и *двоичным кодом*? Запишите все цифры десятичной системы счисления во взвешенных двоично-десятичных кодах 8421, 2421, 5421 и $84\bar{2}\bar{1}$ (здесь $\bar{2}$ и $\bar{1}$ — это отрицательные числа -2 и -1). Убедитесь в том, что код Айкена является самодополняющим.

9. Ознакомьтесь с основными целочисленными арифметическими инструкциями, поддерживаемыми центральными процессорами семейства x86. В частности, найдите в документации принцип действия и соответствующие машинные коды для тех инструкций, ассемблерные мнемоники которых перечислены здесь: `add`, `adc`, `inc`, `sub`, `sbb`, `dec`, `neg`, `mul`, `imul`, `div`, `idiv`.

10. Проанализируйте работу фрагмента программы, исходный код которого приведён в разделе 1.3. Ознакомьтесь с инструкциями центральных процессоров семейства x86, поддерживающими двоично-десятичную и ASCII-арифметику. Более конкретно, — найдите в документации принцип действия и соответствующие машинные коды для инструкций с ассемблерными мнемониками: `daa`, `das`, `aaa`, `aas`, `aam`, `aad`. В связи с ASCII-арифметикой полезно также проанализировать следующий фрагмент исходного кода программы.

```
1 mov ax, 3137
2 mov bx, 3332
3 add ax, bx
4 aaa
```

Источники

1. Петцольд Ч. Код. Тайный язык информатики.
2. Patt Y.N., Patel S.J. Introduction to Computing Systems: From Bits & Gates to C/C++ & Beyond.
3. Брайант Р.Э., О'Халларон Д.Р. Компьютерные системы: архитектура и программирование (Computer Systems: A Programmer's Perspective).
4. Керниган Б., Ритчи Д. Язык программирования C.
5. Hyde R. The Art of Assembly Language.
6. Марек Р. Ассемблер на примерах. Базовый курс.
7. Online Binary-Decimal Converter.

8. Cheung Sh.Y. Computer Architecture/Machine Level Programming: Excess 2^{n-1} encoding.

9. Tech Help Manual. 8088/286/386/486/Pentium Instruction Set: Arithmetic and Binary Logic Instructions.

10. Tech Help Manual. 8088/286/386/486/Pentium Instruction Set: Instruction Set Matrix.