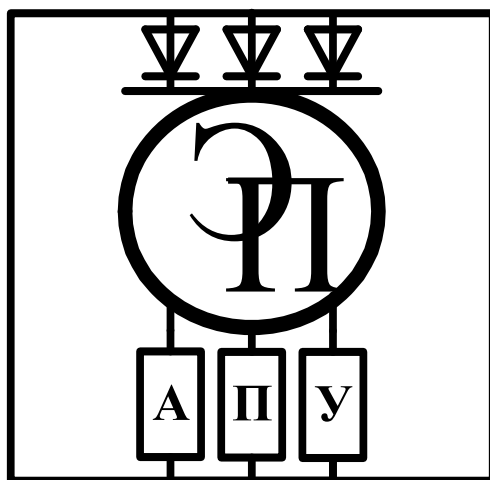


МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Электропривод и автоматизация промышленных установок»

ОСНОВЫ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ

*Методические рекомендации к самостоятельной работе
для студентов специальности 6-05-0713-04 «Автоматизация
технологических процессов и производств»
заочной формы обучения*



Могилев 2026

УДК 004.2
ББК 32.973.26-04
О75

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Электропривод и автоматизация промышленных установок» «4» июня 2026 г., протокол № 10

Составитель О. А. Капитонов

Рецензент Н. В. Герасименко

Методические рекомендации к самостоятельной работе предназначены для студентов специальности 6-05-0713-04 «Автоматизация технологических процессов и производств» заочной формы обучения.

Учебное издание

ОСНОВЫ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ

Ответственный за выпуск

А. С. Коваль

Корректор

И. В. Голубцова

Компьютерная верстка

Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 26 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

© Белорусско-Российский
университет, 2026

Содержание

1 Самостоятельная работа № 1. Арифметические основы вычислительной техники	4
2 Самостоятельная работа № 2. Представление и обработка информации в микропроцессорных системах	9
3 Самостоятельная работа № 3. Логические основы вычислительной техники	13
Список литературы	16

1 Самостоятельная работа № 1. Арифметические основы вычислительной техники

Цель работы: овладеть методикой перевода чисел и выполнения арифметических операций над ними в разных системах счисления.

Задание

Для задач, выданных преподавателем, выполнить перевод чисел, а также произвести над ними ряд арифметических операций.

1.1 Системы счисления

Микропроцессор может обрабатывать только информацию, представленную в числовой форме. Система счисления – это способ записи чисел с помощью заданного набора специальных знаков (цифр).

Запись числа в некоторой системе счисления называется кодом числа. В общем случае число записывается следующим образом:

$$X = a_n a_{n-1} \dots a_0. \quad (1.1)$$

Отдельную позицию в записи числа принято называть разрядом, а номер позиции – номером разряда. Количество разрядов в записи числа – это разрядность, она совпадает с длиной числа.

Если при записи числа значимость («вес») цифры зависит от местоположения (позиции), занимаемого в числе, то такая система счисления называется позиционной. Произвольное число X в позиционной системе с основанием q в общем виде можно записать в виде полинома

$$X_q = \sum_{i=-m}^n a_i q^i = a_n q^n + a_{n-1} q^{n-1} + \dots + a_0 q^0 + a_{-1} q^{-1} + \dots + a_{-m} q^{-m}, \quad (1.2)$$

где a_i – разрядный коэффициент, $a_i = 0 \dots q - 1$;
 q^i – весовой коэффициент.

Число q называется основанием системы счисления. Оно равно количеству различных знаков или символов, используемых для изображения цифр в данной системе. Разряды с положительными степенями q образуют целую часть числа X_q , а с отрицательными – дробную. Цифры a_n и a_{-m} являются старшим и младшим разрядами числа соответственно.

В зависимости от целей применения используют различные системы счисления: двоичную, восьмеричную, десятичную, шестнадцатеричную. В двоичной системе всего две цифры – 0 и 1, в десятичной – 10 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), в восьмеричной – 8 (0, 1, 2, 3, 4, 5, 6, 7), в шестнадцатеричной – 16 (в качестве первых 10 из 16 шестнадцатеричных цифр взяты привычные цифры 0, 1, ..., 9, а вот в качестве остальных шести цифр используют первые буквы латинского алфавита: A, B, C, D, E, F). Чтобы различать, в какой системе

счисления записано число, рядом с ним в виде индекса (записанного в десятичной системе) указывается основание системы счисления.

Пример записи чисел в различных системах счисления:

$$1001,01_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}$$

$$15,4_{10} = 1 \cdot 10^1 + 5 \cdot 10^0 + 4 \cdot 10^{-1}$$

$$A9,5_{10} = 10 \cdot 16^1 + 9 \cdot 16^0 + 5 \cdot 16^{-1}$$

В связи с использованием различных систем счисления при обработке данных на ЭВМ возникает необходимость перевода чисел из одной системы счисления в другую. Существует несколько правил перевода чисел. Сформулируем некоторые из них.

Правило 1. При переводе целого десятичного числа в систему с основанием q его необходимо последовательно делить на q до тех пор, пока не останется остаток, меньший q . Число в системе с основанием q записывается как последовательность остатков от деления, записанных в обратном порядке, начиная с последнего. Деление выполняется по правилам той системы счисления, в которой записано исходное число.

Например, переведем число 95 из десятичной системы счисления в двоичную и восьмеричную (рисунок 1.1).

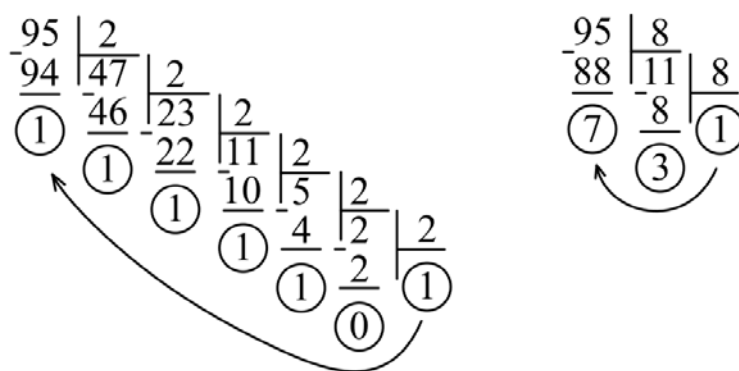


Рисунок 1.1 – Пример перевода числа из одной системы в другую

Получим $95_{10} = 1011111_2 = 137_8$.

Правило 2. При переводе правильной десятичной дроби в систему счисления с основанием q необходимо сначала саму дробь, а затем дробные части всех последующих произведений последовательно умножать на q , отделяя после каждого умножения целую часть произведения. Число в новой системе счисления записывается как последовательность полученных целых частей произведения. Умножение производится до тех пор, пока дробная часть произведения не станет равной нулю. Это значит, что сделан точный перевод. В противном случае перевод осуществляется до заданной точности.

Например, переведем число 0,28125 из десятичной системы счисления в восьмеричную:

$$\begin{array}{r|l}
 & x \\
 0,28125 & 8 \\
 2\ 25 & 8 \\
 2\ 0 &
 \end{array}$$

Получим $0,28125_{10} = 0,22_8$.

Правило 3. При переводе числа из любой системы счисления в десятичную удобно пользоваться разложением числа в виде многочлена по степеням основания.

Например, переведем число 111011_2 в десятичную систему счисления:

$$111011_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 59_{10}$$

Правило 4. При переводе числа из восьмеричной или шестнадцатеричной системы счисления в двоичную каждую восьмеричную или шестнадцатеричную цифру заменяют соответствующей триадой (для восьмеричной системы) или тетрадой (для шестнадцатеричной системы).

Например, переведем числа $537,1_8$ и $1A3,F_{16}$ в двоичную систему счисления:

$$537,1_8 = 101\ 011\ 111,001_2$$

$$1A3,F_{16} = 1\ 1010\ 0011,1111_2$$

Правило 5. При переводе из двоичной системы счисления в восьмеричную или шестнадцатеричную исходное число разбивают (вправо и влево от запятой) на триады (для восьмеричной) или тетрады (для шестнадцатеричной) и каждую полученную триаду или тетраду заменяют соответствующей ей цифрой в восьмеричной или шестнадцатеричной системе.

Например, переведем число $10010011111,1110011_2$ в шестнадцатеричную систему счисления:

$$10010011111,1110011_2 = 0100\ 1001\ 1111,1110\ 0110 = 49F,E6_{16}$$

В таблице 1.1 приведено соответствие кодов некоторых чисел в различных системах счисления.

Таблица 1.1 – Соответствие кодов некоторых чисел в различных системах счисления

Двоичный	Восьмеричный	Десятичный	Шестнадцатеричный
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6

Окончание таблицы 1.1

Двоичный	Восьмеричный	Десятичный	Шестнадцатеричный
111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

1.2 Арифметические операции в позиционных системах счисления

Рассмотрим основные арифметические операции: сложение, вычитание, умножение и деление. Правила выполнения этих операций в десятичной системе хорошо известны: сложение, вычитание, умножение столбиком и деление углом. Эти правила применимы и ко всем другим позиционным системам счисления.

Сложение чисел выполняется поразрядно, начиная с младшего разряда. Получившееся в результате число не должно превышать основание системы счисления. Поэтому если при сложении возникает избыток, то он переносится влево и учитывается при сложении чисел в следующем разряде. При этом результат сложения уменьшается на основание системы счисления.

Например, при сложении двух чисел X и Y в двоичной системе счисления коды суммы C и переноса Π формируются в соответствии с таблицей 1.2.

Таблица 1.2 – Формирование кодов суммы и переноса

X	Y	Π	C
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Пример 1 – Сложим числа 0111_2 и 0101_2 :

$$\begin{array}{r} 0111 \\ +0101 \\ \hline 1100 \end{array}$$

Ответ: 1100_2 .

Аналогичные правила действуют и для других чисел, представленных в других системах счисления.

Пример 2 – Сложим числа 15 и 6 в шестнадцатеричной системе счисления:

$$15_{10} + 6_{10} = F_{16} + 6_{16} = 15_{16} = 25_8 = 10101_2 = 21_{10}$$

Ответ: 15_{16} .

Для проверки преобразуем полученные суммы к десятичному виду:

$$10101_2 = 2^4 + 2^2 + 2^0 = 16 + 4 + 1 = 21$$

$$25_8 = 2 \cdot 8^1 + 5 \cdot 8^0 = 16 + 5 = 21$$

$$15_{16} = 1 \cdot 16^1 + 5 \cdot 16^0 = 16 + 5 = 21$$

Вычитание также производится поразрядно, начиная с младшего разряда. При вычитании большего числа из меньшего необходимо произвести заем из старшего разряда, что равносильно добавлению числа, равного основанию системы счисления, к уменьшаемому числу. Заем при этом уменьшает следующий разряд на 1.

Пример 3 – Вычтем единицу из чисел 100_2 , 100_8 и 10_{16} :

$$\begin{array}{r} 100 \\ -001 \\ \hline 011 \end{array} \quad \begin{array}{r} 100 \\ -001 \\ \hline 077 \end{array} \quad \begin{array}{r} 10 \\ -01 \\ \hline 0F \end{array}$$

Ответ: $100_2 - 1_2 = 11_2$, $100_8 - 1_8 = 77_8$, $10_{16} - 1_{16} = F_{16}$.

Выполняя умножение многозначных чисел в различных позиционных системах счисления, можно использовать обычный алгоритм перемножения чисел в столбик, но при этом результаты перемножения и сложения однозначных чисел необходимо заимствовать из соответствующих рассматриваемой системе таблиц умножения и сложения.

Ввиду чрезвычайной простоты таблицы умножения в двоичной системе умножение сводится лишь к сдвигам множимого и сложениям.

Пример 4 – Перемножим числа 101101_2 и 101_2 :

$$\begin{array}{r} 101101 \\ \times \quad 101 \\ \hline 101101 \\ 000000 \\ 101101 \\ \hline 11100001 \end{array}$$

Ответ: $101101_2 \cdot 101_2 = 45_{10} \cdot 5_{10} = 225_{10} = 11100001_2$.

Деление в любой позиционной системе счисления производится по тем же правилам, как и деление углом в десятичной системе. В двоичной системе деление выполняется особенно просто, ведь очередная цифра частного может быть только нулем или единицей.

Пример 5 – Разделим число 101101_2 на число 101_2 :

$$\begin{array}{r} 101101 \overline{) 101} \\ \underline{101} \\ 0101 \\ \underline{101} \\ 0 \end{array}$$

2 Самостоятельная работа № 2. Представление и обработка информации в микропроцессорных системах

Цель работы: изучить способы представления некоторых типов информации в микропроцессорных устройствах.

Задание

Записать карту памяти микропроцессорной системы для данных, полученных от преподавателя.

2.1 Общие сведения о представлении информации

Микропроцессор в основных современных приложениях может обрабатывать числовую, текстовую, графическую, звуковую и видеоинформацию. Все эти виды информации для процессора представляются в двоичном коде, т. е. используется алфавит мощностью два (всего два символа 0 и 1). Связано это с тем, что удобно представлять информацию в виде последовательности электрических импульсов: сигнал отсутствует (0), сигнал есть (1). Такое кодирование принято называть двоичным, а сами логические последовательности нулей и единиц – машинным языком.

Таким образом, каждая цифра машинного двоичного кода несет количество информации, равное одному биту. Будучи объединенными, биты формируют двоичное сообщение.

2.2 Представление числовой информации

Числа – основной вид информации, с которой работает микропроцессор. Значения физических величин, измеряемых системами управления, представляют собой числа.

Для записи целых чисел используется либо двоичная система счисления, либо двоично-десятичная форма записи. Как было указано ранее, в двоичной

системе счисления числа записываются при помощи последовательности нулей и единиц (рисунок 2.1).

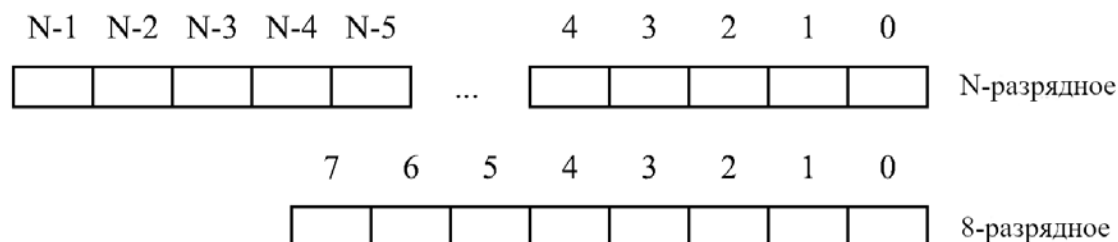


Рисунок 2.1 – Двоичные числа

Как было указано ранее, каждый разряд в таком числе принимает значение 0 или 1. Чем большее количество разрядов используется для записи числа, тем большее значение оно может принимать. Используя двоичную систему счисления, N-разрядное слово позволяет записать 2^N различных значений. Например, 8-разрядное слово может принимать 256 различных значений (от 0 до 255).

Двоично-десятичный код – форма записи рациональных чисел, когда каждый десятичный разряд числа записывается в виде его четырёхбитного двоичного кода. То есть для записи 3-разрядного десятичного числа понадобится 12 двоичных разрядов. Например, число 521 будет записано как 0101 0010 0001₂:

$$\begin{array}{ccc} 5 & 2 & 1 \\ 0101 & 0010 & 0001 \end{array}$$

Для записи отрицательных чисел используются следующие способы [1].

1 Введение дополнительного значащего разряда. В этом случае к положительному числу в старшем разряде вводится дополнительный двоичный разряд, указывающий на знак «0» – число положительное, «1» – отрицательное. Так, на рисунке 2.2 представлены 8-разрядные числа, старший разряд которых не имеет веса, а указывает на знак числа. То есть при дальнейшей обработке он должен быть отброшен.

$$\left\{ \begin{array}{l} 00000101 = 5 \\ 00000000 = 0 \\ 10000000 = 0 \\ 10000101 = -5 \end{array} \right.$$

Рисунок 2.2 – Отрицательные числа со знаковым разрядом

2 Представление в дополнительном коде. В этом случае дополнительный код для отрицательного числа можно получить инвертированием его двоичного модуля (первое дополнение) и прибавлением к инверсии единицы (второе дополнение) либо вычитанием числа из нуля. Так, например, 8-разрядный

дополнительный код числа 5 может быть найден следующим образом:

$$\begin{array}{r}
 00000101_2 \\
 +11111010_2 \\
 \hline
 11111011_2
 \end{array}$$

Как видно, при таком способе представления старший разряд отрицательного числа равен 1 и указывает на знак числа так же, как и в предыдущем способе. Однако отличием является то, что старший разряд в дополнительном коде имеет вес, хоть он и является отрицательным:

$$11111011 = (-1) \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = -5$$

Дробные числа могут быть представлены двумя способами.

1 С фиксированной запятой, когда для хранения целой части числа используется n -разрядное двоичное слово, а для хранения дробной части – m -разрядное двоичное слово (рисунок 2.3).

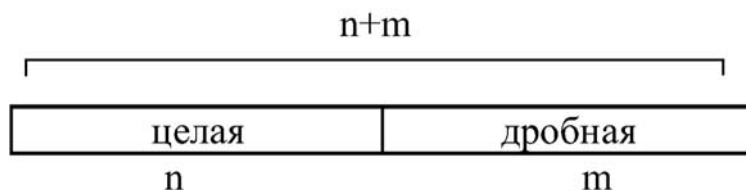


Рисунок 2.3 – Дробное число с фиксированной запятой

2 С плавающей запятой, когда для хранения мантиисы числа используется n -разрядное двоичное слово (с учетом знакового разряда), а для порядка – m -разрядное двоичное слово (с учетом знакового разряда) (рисунок 2.4).

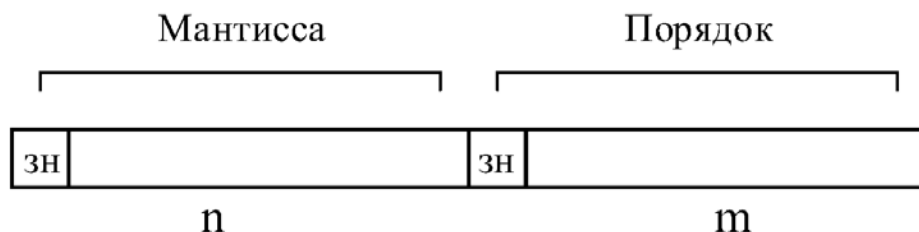


Рисунок 2.4 – Дробное число с плавающей запятой

Рассмотрим пример представления дробного числа с фиксированной запятой, при записи которого используются восемь разрядов для целой и восемь разрядов для дробной части. Для записи такого числа процессору понадобится 16 разрядов, фиксированная запятая расположена между 7-м и 8-м разрядами (рисунок 2.5).

Например, число $45,5_{10}$ после перевода в двоичную систему счисления

принимает вид 101101_2 . Полученное число можно дополнить нулями, не имеющими веса – в старших разрядах для целой части и в младших для дробной, в результате чего число примет вид $00101101,10000000_2$. Целая часть этого числа будет записана в старших восьми разрядах, дробная – в младших (рисунок 2.6).

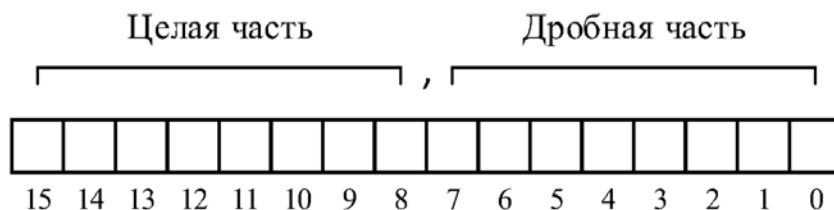


Рисунок 2.5 – 16-разрядное дробное число с фиксированной запятой

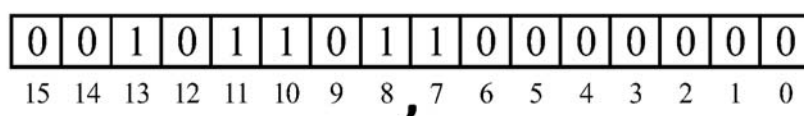


Рисунок 2.6 – 16-разрядное число 45,5 с фиксированной запятой

При сохранении в памяти символы «.» в память не записываются и число $45,5_{10}$ будет записано как двухбайтное число 0010110110000000_2 .

2.3 Порядок следования байтов

Память микропроцессорных систем, как правило, имеет побайтовую организацию, т. е. каждая ячейка позволяет сохранять один байт или 8-битное двоичное слово. Однако, как было ранее указано, такое слово не может превышать значения 255. Числа больше 255 требуют большего количества ячеек для хранения. Так, например, число 60000_{10} представляется 16-разрядным двоичным словом $1110\ 1010\ 0110\ 0000_2$, для хранения которого требуется 2 байта.

Таким образом, информация может быть представлена в виде последовательности байтов. В том случае, если число не может быть представлено одним байтом, имеет значение, в каком порядке байты записываются в памяти компьютера или передаются по линиям связи. В современной вычислительной технике наиболее распространены два способа порядка следования байтов:

- 1) little-endian – от младшего к старшему;
- 2) big-endian – от старшего к младшему.

При использовании порядка «от младшего к старшему» запись начинается с младшего и заканчивается старшим. Для порядка «от старшего к младшему», наоборот, запись начинается со старшего и заканчивается младшим. На рисунке 2.7 приведен пример размещения байтов в памяти. Из него видно, что для порядка little-endian младшие байты многобайтного числа будут располагаться по младшим адресам, а старшие – в ячейках с большим адресом. Для big-endian – наоборот.

Например, если необходимо расположить в памяти по адресу 0 4-байтное

число 500_{10} , потребуется 4 ячейки с адресами 0, 1, 2 и 3. Само число в виде 4-байтного (т. е. 32-разрядного) двоичного числа будет представлено как

$$500_{10} = 000000000000000000000001\ 1111\ 0100_2 = 000001F4_{16}$$

Такая последовательность байтов будет размещена в памяти, как указано на рисунке 2.8.

	Big-endian	Little-endian
Адреса+3	Байт 0	Байт 3
Адреса+2	Байт 1	Байт 2
Адреса+1	Байт 2	Байт 1
Адреса+0	Байт 3	Байт 0

Рисунок 2.7 – Пример размещения байтов слова в памяти

а)

Адрес	Данные
00	F4
01	01
02	00
03	00
04	00
05	00

б)

Адрес	Данные
00	00
01	00
02	01
03	F4
04	00
05	00

a – порядок little-endian; b – порядок big-endian

Рисунок 2.8 – Число 500 в памяти

3 Самостоятельная работа № 3. Логические основы вычислительной техники

Цель работы: изучить способы представления некоторых типов информации в микропроцессорных устройствах.

Задание

Для задач, выданных преподавателем, выполнить ряд логических операций.

Основу микропроцессорных устройств составляют элементарные логические схемы, действующие по законам и правилам алгебры логики.

Алгебра логики – раздел математики, изучающий высказывания, рассматриваемые со стороны их логических значений (истинности или ложности)

и логических операций над ними.

Высказывание – это некоторое законченное предложение, о котором можно однозначно сказать, что оно истинно или ложно.

Каждому высказыванию можно приписать одно из двух значений: истинному высказыванию соответствует значение 1, а ложному – 0.

Из нескольких простых высказываний можно составлять сложные высказывания. Для объединения простых высказываний в сложные применяются логические операции: конъюнкция, дизъюнкция, исключающее ИЛИ, отрицание.

Конъюнкция двух высказываний A и B (логическое умножение, операция И) обозначается $A \cdot B$ или $A \wedge B$ (читается A и B). Правило выполнения: результат операции логического умножения является истинным только тогда, когда истинны оба высказывания. Если оба высказывания или хотя бы одно из них ложно, то и $A \wedge B$ тоже ложно. Таблица истинности логической операции конъюнкция приведена в таблице 3.1.

Дизъюнкция двух высказываний A и B (логическое сложение, операция ИЛИ) обозначается $A \vee B$ или $A + B$ (читается A или B). Правило выполнения: результат операции логического сложения является истинным тогда, когда истинно хотя бы одно из исходных высказываний. Таблица истинности логической операции дизъюнкция приведена в таблице 3.2.

Исключающее ИЛИ двух высказываний A и B (сложение по модулю два) обозначается $A \oplus B$. Правило выполнения: результат операции является истинным, если число складываемых единичных битов нечетно, если же их число четно, то результат ложен. Таблица истинности логической операции Исключающее ИЛИ приведена в таблице 3.3.

Отрицание высказывания A (операция НЕ, инверсия) обозначается $\bar{A}$ (читается «не A »). Правило выполнения: инверсия одного значения переменной совпадает с другим ее значением. Таблица истинности логической операции инверсия приведена в таблице 3.4.

Таблица 3.1 – Таблица истинности логической операции конъюнкция

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Таблица 3.2 – Таблица истинности логической операции дизъюнкция

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Таблица 3.3 – Таблица истинности логической операции исключающее ИЛИ

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Таблица 3.4 – Таблица истинности логической операции инверсия

A	$\bar{A}$
0	1
1	0

Кроме того, микропроцессоры могут производить данные логические операции над группой битов, объединенных в сообщение или слово. При этом операция поочередно выполняется над равнозначными разрядами двоичного слова. Примеры выполнения логических операций над двоичными числами приведены в таблице 3.5.

Для того чтобы выполнить любую из приведенных логических операций над числами в других системах счисления (восьмеричной, десятичной, шестнадцатеричной), их необходимо перевести в двоичный вид, выполнить операцию и перевести в исходную систему счисления.

Таблица 3.5 – Примеры выполнения логических операций над двоичными числами

Операция ИЛИ	Операция И	Операция исключающее ИЛИ	Операция инверсия
$\begin{array}{r} 101101 \\ \vee \\ \underline{001110} \\ 101111 \end{array}$	$\begin{array}{r} 101101 \\ \wedge \\ \underline{001110} \\ 001100 \end{array}$	$\begin{array}{r} 101101 \\ \oplus \\ \underline{001110} \\ 100011 \end{array}$	$\begin{array}{r} \overline{1011001} = \\ = 0100110 \end{array}$

Например, для чисел 67_{10} и 114_{10} будут получены следующие результаты логических операций:

$$67_{10} = 0100\ 0011_2$$

$$114_{10} = 0111\ 0010_2$$

$$\begin{array}{r} 01000011 \\ \vee 01110011 \\ \hline 01110011 \end{array}$$

$$01110011_2 = 2^6 + 2^5 + 2^4 + 2^1 + 2^0 = 115_{10}$$

В этом случае работать с числами, записанными в восьмеричной или шестнадцатеричной системах счисления, значительно проще, т. к. для перевода их в двоичный вид необходимо каждую цифру представить в виде триады (для восьмеричной) или тетрады (для шестнадцатеричной). После чего выполнить логическую операцию.

Например, для чисел $A9_{16}$ и 35_{16} логическая операция И будет выполняться следующим образом:

$$\begin{array}{r} A9 = 01000011 \\ 35 = ^01110011 \\ \hline 01000011 \end{array} = 23_{16}$$

Глядя на результат побитовых операций, не сразу можно уловить закономерности в их результате. Поэтому непонятно, зачем нужны такие операции. Однако они находят свое применение. В байтах не всегда хранятся числа. Байт или ячейка памяти может хранить набор флагов (установлен/сброшен), представляющих собой информацию о состоянии чего-либо. С помощью битовых логических операций можно проверить, какие биты в байте установлены в единицу, можно обнулить биты или, наоборот, установить в единицу. Также существует возможность сменить значения битов на противоположные.

Список литературы

- 1 **Гуров, В. В.** Микропроцессорные системы : учеб. пособие / В. В. Гуров. – М. : ИНФРА-М, 2022. – 336 с.
- 2 **Рассадкин, Ю. И.** Микропроцессорная техника. Специальные вопросы проектирования : учеб. пособие / Ю. И. Рассадкин, А. В. Сеницын. – М. : МГТУ им. Н. Э. Баумана, 2016. – 68 с.
- 3 **Огородников, И. Н.** Микропроцессорная техника: введение в Cortex-M3 : учеб. пособие / И. Н. Огородников. – 2-е изд., стер. – М. : Флинта ; Урал. ун-т, 2017. – 116 с.
- 4 **Новиков, Ю. В.** Основы микропроцессорной техники : учеб. пособие / Ю. В. Новиков, П. К. Скоробогатов. – М. : ИНТУИТ, 2016. – 287 с.