Министерство образования РФ
ФГБОУ ВПО «Сибирская государственная автомобильно-дорожная академия (СибАДИ)»
Кафедра автоматизации технологических процессов и электротехники

АЛГОРИТМЫ ПРЕОБРАЗОВАНИЯ ИНФОРМАЦИИ В МИКРОПРОЦЕССОРНЫХ СИСТЕМАХ

Методические указания к лабораторным работам

Составители: А-й А. Руппель, А-р А. Руппель

Омск
Издательство СибАДИ
2011

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Алгоритмы преобразования информации - основа программирования микропроцессоров независимо от их структуры и длины разрядной сетки. Знание алгоритмов позволяет специалистам легко ориентироваться в уже разработанном программном обеспечении и применять его для решения конкретных задач.

Предлагаемые методические указания рассчитаны на проведение практических занятий и лабораторных работ при изучении восьмиразрядных микропроцессоров на примере К580. Однако полученные знания и практические навыки легко можно применить как при работе с однокристальными ЭВМ KI8I6, так и с более производительными микропроцессорами серии KI8I0.

В методических указаниях приводятся краткие сведения о системах счисления и кодирования чисел в микропроцессорах, наиболее часто применяемые алгоритмы обработки данных и формирования управляющих сигналов.

Представленное контрольное задание позволяет проверить и закрепить знания по алгоритмизации и практическому программированию на примере решения конкретной задачи, включающей основные операции, которые используются в процессе преобразования и обработки информации.

При изучении вопросов, рассматриваемых в данных методических указаниях, большую пользу окажет дополнительное изучение материалов, представленных в [1-4].

1. КОДИРОВАНИЕ И ФОРМЫ ПРЕДСТАВЛЕНИЯ ЧИСЕЛ В МИКРОПРОЦЕССОРАХ (МП)

1.1. Системы счисления, применяемые в МП

Системы счисления - совокупность приемов и правил изображения чисел цифровыми знаками. Наибольшее распространение получили позиционные системы счисления, в которых значение символа зависит от его позиции (места в ряду цифр) в изображении числа. Позиционная система счисления характеризуется основанием, которое выражается числом символов, используемых для изображения цифр в числе.

В разные исторические периоды развития человечества для подсчётов и вычислений использовались те или иные системы счисления. Например, довольно широко была распространена двенадцатеричная система (дюжина, число месяцев в году). В Древнем Вавилоне существовала шестнадцатеричная система (1 ч = 60 мин, 1 мин = 60 с)

У некоторых африканских племён была распространена пятеричная система счисления, у ацтеков и народов майя – двадцатеричная система. Десятичная система измерения возникла в Индии и затем была завезена арабами в Европу.

В технике, в ЭВМ и микропроцессорах широко используется двоичная система счисления, которая очень удобна в реализации для цифровых схем. Например, любой цифровой сигнал может иметь какое-то определенное значение, либо быть равным нулю:

Представим число 342 в следующей форме.

$$342 = 3\cdot 10^{2} + 4\cdot 10^{1} + 2\cdot 10^{0} \quad . \qquad (1.1)$$

Отсюда видно, что число 10 является основанием системы счисления, которая в данном случае называется десятичной, а величина числа определяется коэффициентами при основании.

Таким образом, в общем виде числа можно представить в виде

$$A = a_{n} \cdot X^{n} + a_{n-1} \cdot X^{n-1} + \ldots + a_{0} \cdot X^{0} \quad . \qquad (1.2)$$

В двоичной системе счисления основанием является число 2. В этом случае для записи используются две цифры: 0 и 1. Возьмем, например, число 12 в десятичной системе счисления и разложим его по степеням числа. Получим

$$12=1\cdot 2^{3}+1\cdot 2^{2} + 0\cdot 2^{1} + 0\cdot 2^{0} \quad . \qquad (1.3)$$

Число 12 в двоичной системе счисления запишется следующим образом:

$$1100_{(2)} = 12_{(10)} .$$

Перевод числа из десятичной системы счисления в двоичную производится методом последовательного деления числа на 2 до тех пор, пока частное от деления не станет равным 1. Числа в двоичной системе счисления записываются в виде остатков от деления, начиная с последнего частного, справа налево.

Перевод дробного десятичного числа в двоичную систему осуществляется в два этапа: вначале переводится целая часть числа, затем дробная. Дробная часть переводится путём последовательного умножения дробной части на два. Двоичное число записывается в виде частей чисел, полученных при умножении только дробной части, начиная сверху после запятой. При этом задаётся точность выражений.

Все технические устройства в микропроцессорах реализованы на основе двоичной системы счисления, однако человеку работать с двоичными числами неудобно из-за большого числа разрядов. У человека больше развито образное мышление, т. е. ему легче вспомнить небольшое количество разнообразных цифр, чем группу нулей и единиц. Поэтому наибольшее распространение получили системы счисления по основаниям 16, 10, 8 согласно табл. 1.1.

Шестнадцатеричная система счисления использует 16 символов. Шестнадцатеричное число формируется из двоичного путём объединения отдельных битов двоичного числа по четыре вправо и влево от дробной точки. Таким образом, выделяются тетрады, обозначающие двоичные эквиваленты шестнадцатеричных символов (см. табл. 1.1), которые записываются в виде десяти цифр и первых шести букв латинского алфавита.

Двоично-десятичная форма записи десятичных чисел (BCD) часто используется в интерфейсах измерительных приборов и индикаторах. С целью обработки результатов измерений в микропроцессорных системах без дополнительного преобразования в двоичную систему счисления используется BCD-форма записи чисел, а в состав МП вводится дополнительное устройство, называемое '"десятичный корректор". Кроме того, в состав системы команд МП вводится дополнительная арифметическая команда "десятичная коррекция", которая обычно помещается в программе сразу же после команды двоичного счисления и позволяет обрабатывать двоично-десятичные числа по правилам лестничной арифметики на имеющимся в МП двоичном сумматоре.

Алгоритм работы команды десятичной коррекции следующий. Начиная с младшей тетрады проверяется её код, если он больше девяти или был

установлен признак переноса из этой тетрады, то к ее содержимому добавляется код числа 6 и так далее до старшей тетрады.

Восьмеричная запись двоичных чисел используется в некоторых типах микропроцессоров. Восьмеричная система содержит 8 цифр от 0 до 7. Формируется из двоичного числа путем объединения по три бита влево и вправо до десятичной точки. Каждая группа из трех двоичных разрядов представляет одну восьмеричную цифру (см. табл. 1.1).

Таблица 1.1

Коды чисел в различных системах счисления

Десятичное (D)	Двоичное (B)	Восьмеричное (Q)	Шестнадцатеричное (H)	Двоично-десятичное (BCD)
0	0	0	0	0000
I	1	I	I	0001
2	10	2	2	0010
3	11	3	3	0011
4	100	4	4	0100
5	101	5	5	0101
6	110	6	6	0110
7	111	7	7	0111
8	1000	10	8	1000
9	1001	11	9	1001
10	1010	12	A	1 0000
11	1011	13	B	1 0001
12	1100	14	C	1 0010
13	1101	15	B	1 0011
14	1110	16	E	1 0100
15	1111	17	F	1 0101
16	10000	20	10	1 0110
64	1000000	100	40	0110 0100
100	1100100	144	64	0001 0000 0000

1.2. Формы представлений чисел

Числа в МП могут быть представлены в двух формах: с фиксированной точкой или с плавающей. При этом используется определенное количество двоичных разрядов. Суммарное число двоичных разрядов, обрабатываемых параллельно в МП, называется длиной разрядной сетки микропроцессора.

При представлении чисел с фиксированной точкой положение точки фиксируется в определенном месте разрядной сетки МП. Обычно подра-

зумевается, что точка находится или перед старшим разрядом, или после младшего. В первом случае это дробные числа, которые по абсолютному значению находятся в диапазоне $1 - 2^{-n} \geq |x| \geq 0$, где n - число разрядов. Если точка фиксирована после младшего разряда, то целые числа могут быть представлены в диапазоне $2^n - 1 \geq |x| \geq 0$. Если значения чисел превышают верхнюю границу диапазона, то происходит переполнение разрядной сетки.

Достоинство формы представления чисел с фиксированной точкой в том, что это приводит к значительному снижению аппаратных затрат в микропроцессоре, повышению быстродействия при выполнении арифметических операций. Однако при подготовке алгоритмов работы МП необходимо учитывать диапазон измерения величин, используемых в процессе решения, чтобы с помощью подбора масштабов исключить переполнение разрядной сетки, знак числа кодируется в старшем бите разрядной сетки. В этом разряде 0 соответствует плюсу, а 1 - минусу. Формат чисел с фиксированной точкой используется для выполнения операций над кодами адресов, а также в подпрограммах, реализующих арифметические операции с числами, представленными в форме с плавающей точкой.

Для расширения диапазона представления чисел в микропроцессорных системах применяют форму представления чисел с плавающей точкой. В этом случае число записывается в разрядную сетку в виде двух групп цифр. Одна группа (обычно один байт) соответствует порядку числа *P*, другая (обычно два байта) - мантиссе *M*. Для изображения чисел с плавающей точной используются "показательная" форма записи $X=M \cdot 2^P$, причем порядок *P* - целое число, мантисса *M* - дробное, находящееся в диапазоне 0,5.

Знак мантиссы соответствует знаку числа, порядок определяет реальное положение пробной точки в числе. Обработка чисел в форме с плавающей точкой в МП-системах обычно реализуется в виде набора подпрограмм, что существенно снижает скорость вычислений. Аппаратная реализация операций с плавающей точкой значительно усложняет реализацию микропроцессорной системы.

При сравнительном анализе форм представления чисел необходимо четко усвоить, что точность числа определяется только длиной мантиссы. Введение порядка обеспечивает только расширение диапазона представления чисел, но точность его при ограниченной длине слова снижается.

1.3. Кодирование чисел и символов в МП

Исходные данные в МП могут быть представлены в виде чисел с соответствующими знаками или кодами символов текстовой информации. МП

не имеет специальных средств для разделения кодов чисел и символов, поэтому необходимо знать системы их кодирования и различать эти виды информации на этапе составления алгоритмов ее обработки.

Исходные данные могут быть представлены положительными и отрицательными числами. Для кодирования чисел применяют коды: прямой, обратный, дополнительный, которые могут быть простыми и модифицированными.

Коды положительных чисел при всех видах кодирования одинаковы, так как в старшем знаковом бите находится 0 -признак положительного числа. Кодирование отрицательных чисел зависит от структуры построения арифметического устройства и определяется разработчиком микропроцессорной системы.

В прямом коде старший бит определяет знак числа (1 соответствует отрицательному числу), остальные разряды представляют код модуля (абсолютной величины) отрицательного числа.

Основной недостаток такого кодирования заключается в том, что требуется сложный алгоритм реализации арифметического устройства, реализующий положительные и отрицательные коды чисел.

Представление отрицательных чисел в обратном коде позволяет складывать как отрицательные числа, так и положительные по единому алгоритму, включающему, кроме непосредственно операции сложения, дополнительную операцию циклического переноса признака переполнения разрядной сетки МП.

Для получения обратного кода отрицательного двоичного числа необходимо проинвертировать каждый бит модуля этого числа. Представление отрицательных чисел в дополнительном коде позволяет заменить операцию вычитания выполнением только операций сложения. Дополнительный код отрицательного двоичного числа образуется путём формирования обратного кода и прибавления к младшему разряду единицы.

При сложении чисел в МП могут получаться числа, которые по абсолютной величине больше допустимого значении, что ведёт к искажению результатов вычислений. Для немедленного обнаружения переполнения разрядной сетки в ЭВМ применяются специальные схемы, фиксирующие такие случаи. Наиболее распространенный способ такого контроля - введение модифицированных кодов. При этом бит, находящийся справа от знакового разряда, отводится для дублирования признака знака числа. Если этот контрольный бит и бит знака не совпадают, то формируется сигнал прерывания работы ЭВМ. С помощью специальной программы имеется возможность корректировать результат после возникновения переполнения.

МП предназначены для обработки не только цифровой, но и текстовой информации, которая может содержать буквы, цифры, различные симво-

лы. Наиболее широко распространены коды КОИ-7, КОИ-8 (ASC 11), представленные в прил. 1 и прил. 2.

Код КОИ-7 позволяет кодировать всего 128 символов, в том числе 32 буквы русского алфавита, 26 букв латинского алфавита, 10 цифр и служебные символы.

Для букв латинского алфавита справедлив весовой принцип кодирования символов, при котором веса кодов букв увеличиваются на единицу в алфавитном порядке. Если необходимо расположить список фамилий в алфавитном порядке, то эта операция может быть выполнена путем сравнения кодов двоичных чисел, соответствующих буквам алфавита. Для букв русского алфавита весовой принцип несправедлив.

Код КОИ-8 позволяет кодировать 256 символов, в том числе 31 прописную и 32 строчные буквы русского алфавита, 26 прописных и 26 строчных букв латинского алфавита, 10 цифр и 22 служебных знака. Коды символов в диапазоне 21-5 соответствуют одним и тем же символам для обеих систем кодирования.

Для формирования кода числа из кода символа этого числа необходимо вычесть код 30H. Диапазон кодов 00-20H предназначен для управления действиями устройств, участвующих в передаче символов.

2. АЛГОРИТМЫ СЛОЖЕНИЯ И ВЫЧИТАНИЯ ЧИСЕЛ В МИКРОПРОЦЕССОРЕ

2.1. Алгоритмы сложения и вычитания чисел с фиксированной точкой

Сложение и вычитание однобайтовых кодов в восьмиразрядных МП не представляют особых сложностей, так как эти операции могут быть реализованы соответствующими командами МП.

Алгоритмы сложения и вычитания повышенной точности основаны на представлении чисел кодом длиной 2 и более байтов.

Сложение и вычитание выполняются последовательно, начиная с младших байтов чисел, причем признак переноса обнуляется только при сложении младших байтов чисел.

Рассмотрим примеры программ с многобайтными числами. При этом используются следующие символические имена и адреса:

FIRST	- адрес младшего байта первого операнда и адрес результата;
SECOND	-адрес младшего байта второго операнда;
HL, DE	-адресные указатели данных;
N	-длина операндов в байтах;
B	- счетчик байтов (длина операндов).

Программа сложения многобайтных чисел:

```
ADDN:   MVI   B, N          ; инициализация счетчика
        LXI   D, FIRST      ;инициализация указателя
        LXI   H, SECOND     ;инициализация указателя
        XRA   A             ; сброс признака переноса
LOOP:   LDAX  D             ; загрузка первого операнда
        ADC   M             ; сложение
        STAX  D             ; запись результата
        DCR   B             ; декремент счётчика
        JZ    DONE          ; сложение закончено
        INX   H             ; переход к следующему байту
        INX   D             ;
        JMP   LOOP          ; организация цикла
DONE:                       ; конец
```

Для реализации вычитания необходимо в этой программе заменить команду ADC на CBB. В случае получения отрицательной разности признак переноса устанавливается CY=1.

Для сложения многобайтных BCD- кодированных десятичных чисел в программу ADDN после команды ADC следует ввести команду десятичной коррекции DAA.

Вычитание многобайтных BCD-кодированных десятичных чисел (целых, без знака) рассмотрим на примере следующей программы. Вычитание BCD-чисел выполняется в две фазы: сначала получают дополнения: вычитаемого до 10^{n+1}, а затем складывают полученный код с уменьшаемым. В программе используются следующие символические имена и адреса:

FIRST - адрес уменьшаемого и разности;
DE - указатель адреса текущего байта уменьшаемого;
SECOND - адрес вычитаемого;
HL - указатель адреса текущего байта вычитаемого;
N - длина операндов в байтах;
B - счетчик байтов.

Программа DSUBN:

```
DSUBN:  MVI   B, N          ; загрузка счётчика и
        LXI   D, FIRST      ;указателя адресов операндов
        LXI   H, SECOND     ;
        SCT                 ; установка CY=I
LOOP:   MVI   A,99H         ; загрузка девяток в тетрады
        ACI   0             ; учет признака переноса
        SUB   M             ; дополнение вычитаемого
        XCHG                ; обмен указателей
        ADD   M             ; сложение с уменьшаемым
        DAA                 ; коррекция
```

```
        MOV     M, A            ; запоминание разности
        XCHG                    ; восстановление указателей
        DCR     B               ; вычитание окончено
        JZ      DONE            ; выход из программы
        INX     D
        INX     H
        JMP     LOOP
DONE:   …       …               ; конец.
```

Для представления BCD-чисел со знаком вводится тетрада знака (0000 - положительные числа, 1001- отрицательные). Правило получения десятичного BCD-дополнительного кода отрицательного числа заключается в вычитании его из числа, состоящего из одних девяток, и инкрементировании результата.

Если по завершении программы признак переноса равен нулю, то это свидетельствует о получении отрицательного результата.

2.2. Сложение и вычитание чисел с плавающей точкой

Представление чисел с плавающей точкой требует введения двух специальных операций: нормализации и денормализации кодов чисел. Операция нормализации заключается в том, что число сравнивают со значениями 2^0 и 2^{-1}, а если оно находится за пределами этого диапазона, то код сдвигают вправо или влево с соответствующим увеличением или уменьшением порядка. Эти операции повторяются до тех пор, пока мантисса не окажется в заданном диапазоне.

Операция денормализации заключается в обратном преобразовании, т. е. в зависимости от знака порядка кода мантиссы сдвигается вправо или влево на число подвижных разрядов, равных модулю порядка. Процесс денормализации заканчивается, когда порядок будет ранен нулю. Иногда применяют относительную денормализацию, т.е. сдвигают мантиссу одного из операндов до тех пор, пока порядки этих операндов не выравниваются.

Денормализация необходима для выполнения сложения и вычитания, так как эти операции можно осуществлять только с операндами, имеющими одинаковый порядок. Например,

$$M1 \cdot 2^{p1} + M2 \cdot 2^{p2} - 2^{p1}\left\lfloor M1 + M2 \cdot 2^{(p2 \cdot p1)} \right\rfloor, \qquad (2.1)$$

где $P1 \geq P2$.

В выражении (2.1) умножение на два в степени (*P2-P1*)<=0 соответствует сдвигу второго операнда на соответствующее число разрядов вправо.

Для сдвига при денормализации выбирают всегда тот операнд, который не вызывает переполнения разрядной сетки МП.

После выравнивания порядков применяют подпрограммы по алгоритмам сложения с фиксированной точкой.

При представлении чисел в дополнительном коде (наиболее распространенный способ в МП) сдвиги осуществляются с сохранением знака. Для этого в состав МП включают специальные команды "арифметического" сдвига. Если такие команды отсутствуют, то их реализуют с помощью специальных алгоритмов на основе логических или циклических сдвигов с учетом признака переноса. После выполнения сложения результат опять приводят к нормализованному виду.

Для денормализации порядок меньшего числа необходимо увеличивать до порядка большего числа, а мантиссу меньшего – сдвигать вправо.

3. АЛГОРИТМЫ ПРЕОБРАЗОВАНИЯ ИНФОРМАЦИИ ИЗ ОДНОЙ СИСТЕМЫ СЧИСЛЕНИЯ В ДРУГУЮ

В микропроцессорных системах наибольшее распространение получили шестнадцатеричная (H), восьмеричная (Q), двоичная (B), десятичная (D) системы счисления. Последняя обычно представлена в двоично-десятичной форме (BCD-коды). В основе взаимного преобразования чисел в H,Q, B-системы счисления лежат простейшие операции сдвига и маскирования определенных разрядов кодов двоичного числа. Смысл этих преобразований состоит в перегруппировке разрядов чисел, представленных двоичным кодом. В H-системе двоичные разряды группируются по четыре бита, а в Q-системе - по три. После этого каждая группа заменяется соответствующим символом.

Удобство использования Q, H-систем счисления объясняется их более компактным представлением двоичных чисел и преимуществами образного мышления человека по сравнению с предметным.

3.1. Преобразование из двоичной системы счисления в десятичную

Это наиболее часто используемые операции при реализации приборных интерфейсов микропроцессорных систем, например при выводе результатов на цифровую десятичную индикацию или в процессе преобразования десятичных кодов чисел в коды цифр, представленных в КОИ-7.

Наибольшее распространение получила двоично-десятичная форма представления десятичных чисел с весом разрядов 8-4-2-1. Основной недостаток этой системы кодирования десятичных цифр заключается в том,

что полученные коды являются несамодополняющимися, т. е. прибавлением единицы к коду цифры 9 возникает код 0АН вместо нуля с формированием признака переноса. Этот недостаток компенсируется введением специальной команды десятичной коррекции.

Алгоритм преобразования двоично-кодированного числа в BCD-код заключается в следующем:

1. Обнуляется массив памяти, где будет формироваться BCD-код числа.

2. Двоичный код числа сдвигается на один разряд влево с учетом переноса.

3. Складывается BCD-число само с собой и прибавляется бит переноса (сложение производят с применением команды десятичной коррекции).

4. Повторяется число циклов, начиная со второго пункта, равное числу разрядов двоичного числа.

Таким образом, алгоритм реализует следущее равенство:

$$X_{(BCD)} = (...(a_n) \cdot 2 + a_{n-1}) \cdot 2 + ... + a_1 , \qquad (3.1)$$

где a_n , a_{n-1} , ... , a_1 - разряды двоичного числа.

Достоинство этого алгоритма в том, что отсутствует необходимость хранить в памяти веса разрядов двоичного числа. Они формируются автоматически в процессе преобразования. Допустим, нам необходимо преобразовать число 69Н в BCD-код. В двоичной системе это число имеет вид 01101001В. Принцип работы алгоритма поясняется в табл. 3.1.

Таблица 3.1

Пример преобразования числа 69Н в BCD-код

№ цикла	CY	B–код (69H)	BCD–код	Операция
1	2	3	4	5
1	0 0	01101001 Сдвиг влево 11010010	0000 0000 0000 +0000 0000 0000 0	Сложение (удвоение), перенос (CY)
2	 1	Сдвиг влево 10100100	0000 0000 0000 +0000 0000 0000 + 1	То же
3	 1	Сдвиг влево 01001000	0000 0000 0001 +0000 0000 0001 + 1	То же

Окончание табл. 3.1

1	2	3	4	5
4	0	Сдвиг влево 10010000	0000 0000 0011 +0000 0000 0011 + 0	То же
5	1	Сдвиг влево 00100000	0000 0000 0110 +0000 0000 0110 + 1	То же
6	0	Сдвиг влево 01000000	0000 0001 0011 +0000 0001 0011 + 0	То же
7	0	Сдвиг влево 10000000	0000 0010 0110 +0000 0010 0110 + 0	То же
8			0000 0100 1100 0000 0000 0110	Условие десятичной коррекции
9	1	Сдвиг влево 00000000	0000 0101 0010 +0000 0101 0010 + 1	Сложение (удвоение), перенос (CY)
10			0000 1010 0101 0000 0110 0110	Условие десятичной коррекции
Результат (BCD – число)			0001 0000 0101	

3.2. Преобразование из десятичной системы счисления в двоичную

Этот алгоритм обычно используется в микропроцессорных системах, принимающих информацию от цифровых измерительных приборов, которые на выходной шине имеют, как правило, BCD-кодированную информацию, удобную для восприятия человеком. Преобразование в двоичную систему рекомендуется осуществлять в том случае, если входная информация подвергается сложной математической обработке. Это объясняется тем, что алгоритмы МП-системы имеют более высокое быстродействие при обработке чисел в двоичной системе кодирования. Если же в МП-системе применяются простейшие арифметические преобразования, то их обычно выполняют по алгоритмам обработки BCD-кодов, без дополнительного преобразования в двоичные коды, но это сужает возможный диапазон представления чисел.

Алгоритм преобразования реализуется согласно выражению

$$X_{(b)} = ((...(a_n)\cdot 10 + a_n)\cdot 10 + ... + a_2)\cdot + a_1 , \qquad (3.2)$$

где a_n , a_1 - стерший и младший разряды, десятичного числа, представленного в BCD–коде.

Процесс умножения на 10 обычно заменяется двумя сдвигами, вложением и ещё сдвигом, т.е. $(2 \cdot 2 + 1)\cdot 2 = 10$.

Алгоритм преобразования состоит в последовательности следующих действий:

1. Обнуляется ячейка памяти, где будет накапливаться сумма, формирующая двоичное число.

2. Начиная со старшей, выделяется BCD–кодированная десятичная цифра.

3. Выделенная цифра прибавляется к сумме.

4. Сумма умножается на десять по ранее описанному алгоритму сдвига и сложения.

5. Цикл повторяется, начиная со второго пункта до прибавления последней десятичной цифры.

Принцип работы алгоритма поясняется в табл. 3.2.

Таблица 3.2

Принцип работы алгоритма преобразования BCD-кода (125D) в двоичный

№ цикла	BCD - число	Двоичное число (сумма)	Операция
1	1	0000 0000 + 0001 0000 0001 0000 0100 + 0001 0000 0101 0000 1010	Сложение с BCD–числом 2 сдвига влево суммы Сложение с BCD–суммой Сдвиг влево Результат умножения на 10
2	2	0000 0101 + 0010 0000 1100 0011 0000 + 0010 0011 1100 0011 1100 0111 1000	Сложение с BCD–цифрой 2 сдвига влево суммы Сложение с BCD–цифрой Сдвиг влево Результат умножения на 10
3	5	0111 1000 + 0101	Сложение с BCD-цифрой
Результат – двоичное число		0111 1101	

Преимущество рассмотренного алгоритма заключается в том, что отсутствует необходимость хранения весов разрядов десятичного числа. Они формируются в процессе преобразования.

4. АЛГОРИТМЫ УМНОЖЕНИЯ ЧИСЕЛ В МП

В современных микропроцессорных системах операции умножения и деления чаще всего выполняются аппаратными средствами, но в простейших МП эта операция реализуется на основе подпрограмм. Алгоритм умножения двоичных чисел значительно проще, чем в других системах счисления, и сводится практически к двум операциям - сложения и сдвига.

Существует много различных алгоритмов выполнения операции умножения, имеющих определённые достоинства и недостатки.

Рассмотрим алгоритм умножения двух байтов младшими разрядами вперёд за сдвигом частичной суммы вправо.

Распределение ресурсов МП: D - множимое, C - множитель, BC - двухбайтное произведение, E - счетчик битов.

```
 MUL 88:    MVI     B, 0        ;сброс частичной суммы
            MVI     E, 8        ; загрузка счетчика
MXBIT:      MOV     A, C        ; множитель в АК
            RAR                 ; анализируемый бит в СУ
            MOV     C, A        ; возврат сдвинутого множителя
            DCR     E           ; декремент счетчика битов
            RM                  ; умножение закончено
            MOV     A, B        ; старший байт произведения
            JNC     NOADD       ; бит множителя =0
            ADD     D           ; суммирование множимого
NOADD       RAR                 ; сдвиг частичной суммы
            MOV     B, A        ; возвращение старшего байта
            JMP     NXBIT       ; умножение на следующий бит
```

Для умножения целых однобайтовых чисел с учетом знаков вводятся дополнительные операции формирования и установки знака произведения. Это может быть реализовано с помощью следующей подпрограммы.

```
USUIG:      MOV     A, C        ;
            XRA     D           ; формирование знака
            ANI     80H         ; выделение бита знака
            PUSH    PSW         ; запись в стек
            MOV     A, C        ;
            ANI     7H          ; подавление знаков
            MOV     C, A        ;
            MOV     A, D        ;
```

```
ANI     7H          ;
MOV     D, A        ;
CALL    MUL 88      ; умножение без знака
POP     PSW         ; вызов знака
ORA     B           ; установка знаков произведения
MOV     B, A        ;
RET                 ; возврат
```

Аналогично можно реализовать подпрограммы умножения много байтовых чисел. Кроме того, имеется возможность использовать эти подпрограммы или аппаратное множительное устройство, если, например, двухбайтные числа представить в виде сумм:

$$A = 256 \cdot A1 + A2; \quad B = 256 \cdot B1 + B2, \qquad (4.1)$$

где *A1*, *B1* - старшие байты; *A2*, *B2* - младшие.

Тогда произведение будет

$$C = AB = 256 \cdot 256 \cdot A1 \cdot A2 + 256(A1 \cdot B2 + B1 \cdot A2) + A2 \cdot B2. \qquad (4.2)$$

Такой алгоритм требует четырехкратного обращения к подпрограмме умножения однобайтных чисел. Числовые множители определяют положение байта в результате, длина которого составляет 4 байта.

С целью сохранения точности вычислений результаты произведений должны быть всегда представлены в виде операндов двойной длины по сравнению с сомножителями.

Умножение чисел, представленных в форме с плавающей точкой, выполняют по алгоритму, реализующему следующее тождество:

$$M_1 \cdot 2^{P1} \cdot M_2 \cdot 2^{P2} = M_2 M_2 2^{P1+P2}, \qquad (4.3)$$

где *Ml*, *M2* - мантиссы сомножителей; *P1*, *P2* - порядки.

Очевидно, что для выполнения операции умножения необходимо отдельно перемножить мантиссы и сложить порядки. В случае если сомножители были такими, что произведение вышло за пределы отведённого диапазона для мантиссы, требуется операция нормализации.

5. АЛГОРИТМЫ ДЕЛЕНИЯ ЧИСЕЛ МП

Основу реализации алгоритма деления составляют операции вычитания, сдвига, сравнения и при необходимости восстановления остатка, т.е.

несколько сложнее умножения. В результате выполнения операции деления получают целую часть частного и положительный остаток.

Рассмотрим подпрограмму деления со сдвигом остатков влево и восстановлением остатков. Делимое должно быть помещено в регистр Е, делитель - в Д, частное образуется в регистре Н, положительный остаток - в регистре С.

```
DIV 88:   LXI   H, 8     ;  инициализация счётчика битов
          MVI   C, 0     ; сброс регистра остатков
NEXT      MOV   A, E     ; сдвиг делимого
          RAL            ; влево
          MOV   E, A     ; на один бит
          MOV   A, C     ; сдвиг   остатка
          RAL            ; влево на один бит
          SUB   D        ;  вычитание делителя
          JNC   NOADD; остаток положительный
          ADD   D        ; восстановление остатка
NO ADD:   MOV   C, A     ; запоминание остатка
          CMC            ; образование бита частного
          MOV   A, H     ; запоминание
          RAL            ; очередной
          MOV   H, A     ; цифры частного
          DCR   L        ; декремент счётчика битов
          JNZ   NEXT ; организация цикла
          RET            ; возврат
```

До обращения к этой подпрограмме признак переноса должен быть сброшен (CY =0*).*

При делении делитель последовательно вычитается из делимого и после каждого вычитания анализируется бит займа CY. Если делитель больше той части делимого, из которой он вычитается, то CY= 1. В этом случае цифра частного равна нулю и делитель суммируется с результатом вычитания для восстановления положительного знака остатка.

Если при вычитании звеньев не возникает (CY = 0), то цифра частного равна 1, а результат используется после сдвига на один разряд влево как новое промежуточное делимое.

Операция деления прекращается, если делитель равен нулю, остаток равен нулю или получено число разрядов частного, обеспечивающее требуемую точность.

Для деления чисел с учётом их знаков сначала формируется знак частного и выполняется деление положительных чисел, а затем частному присваивается знак.

```
DISIG:  MOV     A, E     ; формирование
```

```
XRA     D       ; знака
ANI     8 0 H   ; частного и
PUSH    PSW     ; запись в стек
MOV     A, E    ; подавление знаков
ANI     7 F, H  ; делимого
MOV     E, A    ;
MOV     A, D    ;
ANI     7 E H   ; и делителя
MOV     D, A    ;
CALL    DIU 88  ; деление без знаков
POP     PSW     ;
ORA     H       ;
RET             ; выход из программы
```

Для сохранения требуемой точности делимое должно иметь удвоенное число разрядов по сравнению с делителем и частным.

Деление чисел с плавающей точкой производится в соответствии с формулой

$$\frac{\left(M_1 2^{P1}\right)}{\left(M_2 B 2^{P2}\right)} = \left(\frac{M1}{M2}\right) 2^{P1-P2}. \qquad (5.1)$$

Деление мантисс и вычитание порядков осуществляется по алгоритмам с фиксированной точкой.

6. АЛГОРИТМЫ ВЫЧИСЛЕНИЯ ЭЛЕМЕНТАРНЫХ ФУНКЦИЙ В МИКРОПРОЦЕССОРАХ

Чаще всего в МП значения элементарных функций определяются путем разложения функций в степенной ряд. Число членов ряда определяется требуемой точностью вычислений. Причем чем выше требуемая точность, тем больше затрачивается времени для вычисления суммы членов ряда.

Для создания программ вычисления элементарных функций от аргументов, представленных в форме с плавающей точкой, необходимо иметь соответствующую библиотеку подпрограмм. Обычно в МП-системах управления точность вычислений в четыре десятичных знака вполне удовлетворяет для большинства решаемых задач.

Вызов любой подпрограммы из библиотеки производится следующим образом:

1. Загрузить операнд (аргумент функции) в специальную ячейку памяти (называемую псевдосумматором).
2. Вызвать подпрограмму вычисления функции.
3. Читать результат из псевдосумматора.

Для вычисления элементарных функций, кроме разложения в ряд Тейлора, большое распространение получили итерационные методы, наиболее известными из которых являются методы Волдера и Меджитта. К недостаткам этих методов можно отнести большое количество хранимых констант. Достоинствами являются высокое быстродействие алгоритмов, простая организация вычислительного процесса.

Методы Волдера и Меджитта относятся к методам вычисления «цифра за цифрой» и сводятся к выполнению двух этапов, каждый из которых представляет итерационный процесс, состоящий в построении последовательности

$$Y_{i+1} = f\,(Y_i)\,. \qquad (6.1)$$

Первый этап заключается в определении набора операторов ξ_i, значения которых вычисляются по соответствующему закону на основании знака ΔY_i (разности между двумя соседними шагами итерации).

На втором этапе на основании найденного набора значений операторов ξ_i определяется значение вычисляемой функции либо путем суммирования значения функции, вычисленного на предыдущем шаге, с константой одного из видов: $\ln(1+\xi_i \cdot 2^{-i})$ или $\xi_i \cdot \operatorname{arctg}^{-i}$, либо с помощью произведения чисел $(1+\xi_i \cdot 2^{-i})$ в зависимости от вида вычисляемой функции. Если итерационный процесс знакопеременный, то оператор ξ_i принимает значения –1 или +1. При знакопостоянных шагах итерации он принимает 0 или 1.

Вычисление функций по методам Волдера и Меджитта осуществляется по рекуррентным соотношениям, содержащим только операции сдвига и алгебраического сложения. В методе Волдера оба этапа выполняются одновременно и представляют единый процесс. Вычисление по методу Меджитта предполагает последовательное выполнение этапов, т. е. сначала определяется направление поворотов вектора и осуществляется запоминание их в виде чисел $\xi_0, \xi_1, \ldots, \xi_{n-1}$, а затем вычисляются соответствующие координаты.

Точность методов «цифра за цифрой» зависит от числа итераций, которое ограничено разрядной сеткой МП. Число используемых констант также зависит от длины разрядной сетки.

7. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ТИПОВЫХ ФУНКЦИЙ УПРАВЛЕНИЯ

При программировании микропроцессорных контроллеров, применяемых для управлений технологическим оборудованием, часто возникает необходимость использования некоторых типовых процедур управления,

например опрос состояния двоичного датчика, ожидание события, формирование временной задержки и выгодных управляющих сигналов

7.1. Опрос двоичного датчика

Двоичный (бинарный) датчик обычно подключается к одной линии порта ввода МП-системы и может быть замкнут или разомкнут. Этим он формирует контроллер о состоянии объекта, имеющего соответственно два состояния: включен или отключен (открыто или закрыто, вверху или внизу и т. п.).

Обычно когда контакты датчика замкнуты, сигнал на МП-контроллер поступает низкого уровня, если разомкнут - высокого уровня, соответствующего логической 1.

Для определения состояния объекта в некоторой части управляющей программы контроллера опрашивается линия порта ввода, подключенного к соответствующему датчику. Если этот разряд находится в 0, то управление передаётся фрагменту программы, контролирующей включенное состояние объекта, в противном случае выполняется очередная команда программы, соответствующая отключенному состоянию.

Программа, реализующая эту процедуру, может иметь следующий вид:

```
INP:  LDA      0C801H     ; ввод байта АК
      ANI        80H      ; маскирование всех разрядов
                          ; кроме старшего байта
      JZ          ON      ; если объект включен
      LXIH,      OFF      ; если объект отключен
      CALL     0F818H     ; вывод сообщения
      JMP        INP      ; переход в начало
      LXI H,     ONS      ; объект включен
      CALL     0F818H     ; ввод сообщения
      JMP        INP      ; переход в начало
ONS:  DB                  ; «объект включен»  0DH, 00H
OFF:  DB                  ; «объект отключен»  0DH, 00H
      END                 ;
```

В реальном контроллере вместо вызова подпрограммы вывода сообщения на экран формируется необходимое управляющее воздействие. Изменение состояния датчика имитируется в учебной ЭВМ нажатием клавиши F2.

7.2. Ожидание события

Контроллеры технологических объектов работают в реальном масштабе времени, и, следовательно, их функционирование должно опреде-

ляться событиями, происходящими в объекте управления. Чаще всего события в объекте управления фиксируются с использованием двоичных датчиков, например замыкание или размыкание концевого переключателя при перемещении определенного органа объекта управления.

Если требуется по ходу выполнения управляющей программы приостановить формирование каких-либо сигналов управления до тех пор, пока в результате процессов, происходящих в объекте управления, не замкнется контакт датчика, то применяют алгоритмы ожидания.

Программа ожидания поступления сигнала от датчика, имитируемого нажатием клавиши F2 учебной ЭВМ, может быть реализована по следующему алгоритму:

```
WATE:   LXI    H,  WT          ; адрес сообщения
        CALL   0F818H          ; вывод сообщения
LOOP:   LDA    0C801H          ; ввод байта в аккумулятор
        ANI    40H             ; маскирование всех разря-
                               ; дов кроме  D6
        JZ     WATE            ; ожидание сигнала
        LXI    H,  SG          ; от датчика
        CALL   0F818H          ; вывод сообщения
        JMP    LOOP            ; цикл
WT:     DB     «ожидание сигнала»  0DH, 00H
SG:     DB     «сигнал поступил»   0DH, 00H
```

В этих программах обращение к подпрограмме с адресом 0F818H обеспечивает вывод на экран в символьном виде сообщения, адрес которого записывается в регистровую пару HL.

7.3. Формирование управляющего сигнала

Простейшее управляющее воздействие - бинарное (включить или выключить объект). В МП-системе такая операция выполняется путем передачи байта из аккумулятора в порт, отдельные биты которого управляют соответствующим исполнительным механизмом объекта.

Программа включения и выключения объекта может быть реализована в следующем виде:

```
        LXI    H,  STR     ; начальный адрес сообщения
        CALL   0F818H      ; вывод сообщения на экран
ON:     MVI    A,  80H     ; загрузка управляющего
                             кода в АК включения
        STA    0F801H      ; вывод в порт управления кода
        CALL   0F803H      ; ожидание нажатия клавиши
OFF:    MVI    A,  00H     ; код включения объекта
```

```
        STA     0C801H    ; вывод в порт управления кода
        CALL    0C803H    ; ожидание нажатия клавиши
        JMP       ON      ; организация цикла
STR:    DB        «управление знакогенератором»  0DH  00H
```

В этой программе роль объекта выполняет знакогенератор, управляемый старшим битом порта. Подпрограмма с адресом обращения 0F 803H ожидает нажатия любой черной клавиши учебной ЭВМ для выдержки переменного интервала, индуцирующего состояние знакогенератора на экране. Эта программа реализует функцию клавиши F2 компьютера.

7.4. Формирование временной задержки

Реализация временной задержки в МП основана на зацикливании программы и подсчете числа выполненных циклов. Для этого в регистр общего назначения предварительно загружается число, которое в каждом цикле изменяется на единицу. Окончание цикла обычно контролируется по нулевому содержимому используемого регистра. Время задержки при этом определятся числом циклов, умноженным на суммарное время выполнения команд, образующих этот цикл.

Рассмотрим подпрограмму, реализующую временную задержку.

```
TIME    MVI     B, X   ; загрузка числа циклов
COUNT   DCR       B    ; декремент B
        JNZ     CONT   ; организация цикла
        RET              ; возврат в основную программу
```

Для получения требуемой переменной задержки необходимо определить значение числа X, загружаемого в рабочий регистр B. Для этого необходимо по справочнику определить число тактов, за которые выполняется каждая команда, и умножить на длительность одного такта синхронизации, формируемого генератором тактовой частоты МП-системы. Для МП К580 эта частота обычно равна 2 МГц (T – 0,5 мкс).

Тогда можно определить длительность выполнения каждой команды:

```
CALL    TIME      17 тактов (8,5 мкс);
MVI     B, X       7 тактов (3,5 мкс);
DCR     B          5 тактов (2,5 мкс);
JNZ     COUNT     10 тактов (5,0 мкс);
RET               10 тактов (5,0 мкс);
```

Если учитывать, что команды MVI, CALL и RET в этой программе выполняются всего один раз, то общая временная задержка в микросекундах может быть определена по формуле

$$T = (8{,}5 + 5 + 3{,}5) + (2{,}5 + 5) \cdot X \ , \qquad (7.1)$$

где X - число, записанное в регистр B.

Выражая отсюда X и преобразуя его в шестнадцатеричную систему, можно обеспечить требуемую задержку, например 250 мкс:

$$X = \frac{T - 17}{7.5}; \quad X_{250} = \frac{250 - 17}{7.5} \approx 31_{(10)} \ . \qquad (7.2)$$

Таким образом, записав в регистр B код 1FH, можно получить временную задержку 249,5 мкс.

Если полученная точность формирования временного интервала не устраивает разработчика, можно в цикл включить несколько дополнительных холостых команд (например, NOP) и подобрать соответствующее число циклов так, что суммарная задержка будет формироваться с большей точностью.

Формирование управляющих последовательностей импульсов осуществляется путем многократного обращения к подпрограммам временной задержки и вывода управляющих сигналов.

7.5. Формирование псевдослучайного числа

Для контроля работы МП-контроллера, управляющего технологическим процессом, необходимо имитировать различные случайные ситуации. Обычно это осуществляется путём формирования случайных чисел в заданном диапазоне и использования их в качестве сигналов обратной связи от датчиков с наложенным на них шумом.

В контроллерах, использующих алгоритм ожидания какого-либо события, случайное число может быть получено путём выделения младшего байта счётчика циклов ожидания.

Псевдослучайные значения могут быть сформированы также путем записи в регистр любого начального кода, затем выполняются последовательно операции циклического сдвига и сложения по модулю два полученного кода с предыдущим значением.

Кроме этих алгоритмов, можно использовать операции арифметического сложения и выделения маски следующим образом:

```
PRB:    LHLD    P W     ; случайное число
        MVI     C, 16   ; загрузка счётчика
LOOP:   MOV     A, H    ;
        DAP       H     ;
        ANI     60H     ; маска
```

```
        JPE     MT1     ; проверка на нечётность
        INX     H       ;  инкремент
        DCR     C       ; декремент регистра C
MT1:    JNZ     LOOP    ; организация цикла
        SHLD    PW      ; хранение псевдослучайного числа
        RET
PW:     DW 1            ; рабочая ячейка
```

После каждого обращения к этой программе в регистровой паре HL формируется код псевдослучайного числа.

Библиографический список

1. Программная реализация преобразования информации в микропроцессорных системах:: Методические указания указания / Сост. В. В. Петров.–Омск, 1991.

2. Григорьев В. Л. Программное обеспечение микропроцессорных систем. – М.: Энергоатомиздат, 1987. – 228 с.

3. Каган Б. М., Сташин В. В. Основы программирования микропроцессорных устройств автоматики. – М.: Энергоатомиздат, 1987. – 232 с.

4. Майоров В. Г., Гаврилов А. И. Практический курс программирования микропроцессорных систем.– М.: Машиностроение, 1989. – 272 с.

Система команд микропроцессора КР580ИК80А

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	NOP	LXI B,&	STAX B	INX B	INR B	DCR B	MVI B,#	RLC	---	DAD B	LDAX B	DCX B	INR C	DCR C	MVI C,#	RRC	0
1	---	LXI D,&	STAX D	INX D	INR D	DCR D	MVI D,#	RAL	----	DAD D	LDAX D	DCX D	INR E	DCR E	MVI E,#	RAR	1
2	---	LXI H,&	SHLD *	INX H	INR H	DCR H	MVI H,#	DAA	---	DAD H	LHLD *	DCX H	INR L	DCR L	MVI L,#	CMA	2
3	---	LXI SP,&	STA *	INX SP	INR M	DCR M	MVI M,#	STC	---	DAD SP	LDA *	DCX SP	INR A	DCR A	MVI A,#	CMC	3
4	MOV B,B	MOV B,C	MOV B,D	MOV B,E	MOV B,H	MOV B,L	MOV B,M	MOV C,A	MOV C,B	MOV C,C	MOV C,D	MOV C,E	MOV C,H	MOV C,L	MOV C,M	MOV C,A	4
5	MOV D,B	MOV D,C	MOV D,D	MOV D,E	MOV D,H	MOV D,L	MOV D,M	MOV D,A	MOV E,B	MOV E,C	MOV E,D	MOV E,E	MOV E,H	MOV E,L	MOV E,M	MOV E,A	5
6	MOV H,B	MOV H,C	MOV H,D	MOV H,E	MOV H,H	MOV H,L	MOV H,M	MOV H,A	MOV L,B	MOV L,C	MOV L,D	MOV L,E	MOV L,H	MOV L,L	MOV L,M	MOV L,A	6
7	MOV M,B	MOV M,C	MOV M,D	MOV M,E	MOV M,H	MOV M,L	HLT	MOV M,A	MOV A,B	MOV A,C	MOV A,D	MOV A,E	MOV A,H	MOV A,L	MOV A,M	MOV A,A	7
8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADDA	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC M	ADC A	8
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUBA	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUB A	9
A	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A	XRA B	XRA C	XRA D	XRA E	XRA H	XRA L	XRA M	XRA A	A
B	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A	CMP B	CMP C	CMP D	CMP E	CMP H	CMP L	CMP M	CMP A	B
C	RNZ	POP B	JNZ *	JMP *	CNZ *	PUSH B	ADI #	RST 0	RZ	RET	JZ *	---	CZ *	CALL *	ACI #	RST 1	C
D	RNC	POP D	JNC *	OUT N	CNC *	PUSH D	SUI #	RST 2	RC	---	JC *	IN N	CC *	---	SBI #	RST 3	D
E	RPO	POP H	JPO *	XTH L	CPO *	PUSH H	ANI #	RST 4	RPE	PCHL	JPE *	XCH G	CPE	---	XRI #	RST 5	E
F	RP	POP PSW	JP *	DI	CP *	PUSH PSW	ORI #	RST 6	RM	SPHL	JM *	EI	CM *	---	CPI #	RST 7	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

Коды команд КОИ-7 микропроцессора КР580ВМ80А

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	ПУС	НЗ	НТ	КТ	КП	КТМ	ДА	ЗВ	ВШ	ГТ	ПС	ВТ	ПФ	ВК	ВЫХ	ВХ	0
1	АРI	СУ1	СУ2	СУЗ	СТП	НЕТ	СИН	КБ	АН	КН	ЗМ	АР2	РФ	РГ	РЗ	РЭ	1
2	ПРО БЕЛ	!	“	#	$	%	&	‘	(	)	*	+	,	-	.	/	2
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	3
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	4
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	¬	_	5
6	Ю	А	Б	Ц	Д	Е	Ф	Г	Х	И	Й	К	Л	М	Н	О	6
7	П	Я	Р	С	Т	У	Ж	В	Ь	Ы	З	Ш	Э	Щ	Ч	ЗБ	7
8																	8
9																	9
A																	A
B																	B
C																	C
D																	D
E																	E
F																	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	