Практичні заняття № 1.

СИСТЕМИ ЧИСЛЕННЯ ТА ПЕРЕТВОРЕННЯ ЧИСЕЛ З ОДНІЄЇ СИСТЕМИ ЧИСЛЕННЯ В ІНШУ

Цель практических занятий – изучить системы счисления и освоить методику преобразования чисел из одной системы счисления в другую.

Двоичная система счисления

Человеческая деятельность предполагает использование десятичной системы счисления. Десятичная система, или система с основанием 10, содержит 10 цифр (от 0 до 9). Она также характеризуется значением позиции (или весом). В табл.1.1 показано, например, что десятичное число 1327 равно одной тысяче, плюс три сотни, плюс два десятка, плюс семь единиц (1000+300 + 20+7 = 1327).

Таблица 1.1 *Значения позиций десятичных чисел*

Степень основания	10^3	10^2	10^1	10^0
Значения позиций	1000	100	10	1
Двоичные	1	3	2	7
Десятичные	1000	300 +	20 +	7 = 1327

Двоичная система счисления проще десятичной. В ней используются только два символа, что хорошо согласуется с техническими характеристиками цифровых схем, имеющих лишь два устойчивых состояния. Как правило, в качестве символов в двоичной системе служат 0 и 1.

При сравнении записи десятичных и двоичных чисел выясняется, что последние занимают значительно больше позиций, поскольку для их представления используются лишь два символа. В двоичной системе счисления, так же как и в десятичной, каждой позиции (разряду) присвоен определенный вес. Но если в десятичной системе вес равен числу 10 в некоторой степени, то в двоичной системе вместо числа 10 используется число 2. Веса первых 13 позиций (разрядов) цифр двоичного числа имеют следующие значения:

4096 2048 1024 512 256 128 64 32 16 8 4 2 1

2^{12} 2^{11} 2^{10} 2^9 2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2°

В двоичной системе счисления даже сравнительно небольшие числа занимают много позиций. Например:

$$101101_2 = 1x2^5+0x2^4+1x2^3+1x2^2+0x2^1+1\ x2^\circ=45_{10},$$

т. е. двоичное число 101101 имеет ту же величину, что и десятичное число 45. Для удобства идентификации записи двоичных и десятичных чисел в виде нижнего индекса числа записывают 2 и 10 соответственно. Так, например, 101_2 — двоичное число, а 101_{10}-десятичное, причем они выражают разные величины.

Как и в десятичной системе, в двоичной системе для отделения дробной части от целой используется точка *(двоичная точка).* Каждая позиция справа от этой точки имеет свой вес-вес разряда дробной части числа.

Значение веса в этом случае равно основанию двоичной системы, возведенному в отрицательную степень. Такие веса - это дроби 1/2, 1/4, 1/8, 1/16, 1/32 и т.д., которые могут быть записаны как 2^{-1} , 2^{-2} ,2^{-3}, 2^{-4} , 2^{-5} и т.п. Выразим

эти веса через десятичные дроби:

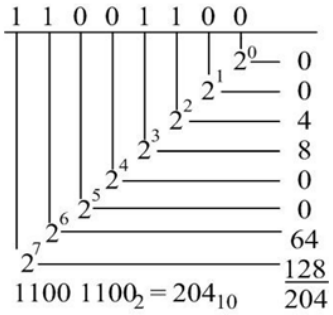
$2^{-1} = 0.5\ 2^{-2} = 0.25\ 2^{-3} = 0.125\ 2^{-4}=0.0625\ 2^{-5}=0.03125\ 2^{-6} = 0.015625$ и т.д.

При записи числа в десятичной системе каждая позиция занята десятичной цифрой, Аналогично при записи двоичного числа каждая позиция занята двоичной цифрой, называемой битом. Говоря о двоичных числах, часто пользуются понятиями наименьший значащий бит (самый младший двоичный разряд) и наибольший значащий бит (самый старший двоичный разряд) по аналогии с наибольшей и наименьшей значащими цифрами десятичного числа. Наименьший значащий бит имеет наименьший вес, а наибольший значащий бит соответственно наибольший. Обычно двоичное число записывается так, что наибольший значащий бит является крайним слева.

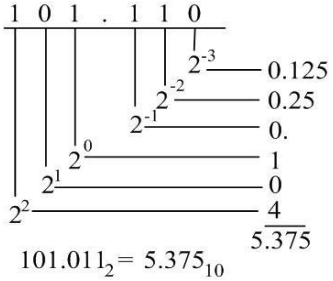
При работе с микро-ЭВМ часто бывает необходимо заменить двоичные числа их десятичными эквивалентами.

Процедура преобразования двоичного числа в десятичное проста: необходимо сложить десятичные веса всех разрядов двоичного числа, в которых содержатся единицы. Продемонстрируем это на следующих примерах.

Пример 1. Преобразование целого двоичного числа 11001100 в десятичное:



Пример 2. Преобразование вещественного двоичного числа 101.011 в десятичное:



Процедура преобразования целых десятичных чисел в двоичные - это частный случай процедуры перевода чисел из одной системы счисления в другую. Предположим, что необходимо преобразовать десятичное число 10 в двоичное. Для этого проделаем следующее.

1. Разделим подлежащее преобразованию число на основание системы счисления, в которой число должно быть представлено. В данном случае 10 следует поделить на 2. При делении на 2 остаток может быть равен 1 или 0. Значение остатка присваивается младшему значащему разряду (МЗР) искомого числа. Для рассматриваемого примера частное равно 5, а остаток - нулю, т.е. 1 - й разряд равен нулю.

2. Результат деления па первом шаге необходимо разделить еще раз на 2. Остаток (0 или 1) используется в качестве значения следующего по значимости разряда. В данном случае частное от деления 5 на 2 равно 2, а остаток, т. е. значение 2-го разряда, равно 1.

3. Результат деления на предыдущем шаге необходимо разделить па 2, а значение остатка присвоить очередному разряду. В данном случае частное равно 1, а остаток равен нулю, т.е. 3-й разряд равен нулю.

4. Шаги описанной процедуры повторяются до тех пор, пока частное, полученное в результате очередной операции деления, не станет равным нулю. Тогда остаток от последнего деления используется в качестве значения старшего значащего разряда (СЗР). В данном случае частное от деления 1 па 2 составляет нуль, а остаток равен 1, поэтому значение 4-го разряда равно 1.

Итак, получено целое двоичное число 1010. Рассмотрим еще два примера преобразования десятичных чисел в двоичные.

Пример 1. Преобразование десятичного числа 57_{10} в двоичное число:

Шаг	Деление	Частное	Остаток
1	57/2	28	1 (МЗР)
2	28/2	14	0
3	14/2	7	0
4	7/2	3	1
5	3/2	1	1
6	1/2	0	1 (СЗР)

Результат: $57_{10} = 1000\ 0110_2$,

Пример 2. Преобразование десятичного числа 134_{10} в двоичное число:

Шаг	Деление	Частное	Остаток
1	134/2	67	0 (МЗР)
2	67/2	33	1
3	33/2	16	1
4	16/2	8	0
5	8/2	4	0
6	4/2	2	0
7	2/2	1	0
8	1/2	0	1 (СЭР)

Результат $-134_{10} = 1000\ 0110_2$

Изложенная процедура применима к преобразованию целых (или целой части) десятичных чисел в двоичные. Для дробных чисел (или дробных частей вещественных чисел) требуется отдельная, хотя и похожая, процедура. Если преобразовать выполнено отдельно для целой и дробной частей числа, то результат получают путем записи двоичных эквивалентов этих частей соответственно слева и справа от двоичной точки.

Процедуру преобразования десятичной дроби в двоичную рассмотрим на

примере преобразования числа 0.375.

1. Преобразование осуществляется умножением дроби на основание системы счисления, в которой дробь должна быть представлена. В данном случае умножаем на 2: 2 х 0.375 = 0.75.

2. Если результат умножения меньше 1, то старшему значащему разряду присваивается значение 0; если больше 1, то присваивается 1. Поскольку 0.75 < 1, то СЗР = 0.

3. Результат предыдущей операции умножения опять умножается на 2. Заметим, что если бы результат предыдущей операции умножения был больше 1, то в данной операции умножения участвовала лишь его дробная часть. В данном случае 2 х 0.75 = 1.5.

4. Если полученный результат меньше 1, то следующему по значимости (ближайшему справа) разряду присваивается значение 0; если равен или больше 1, то присваивается 1. В рассматриваемом примере 1.5 > 1, поэтому значение разряда 2 равно 1.

5 Шаги описанной процедуры повторяются до тех пор, пока либо результат умножения не будет точно равен 1, либо не будет достигнута требуемая точность. В данном случае после выполнения очередного шага результат равен (2 х 0.5=1.0). Поэтому очередному разряду, являющемуся младшим значащим разрядом, присваивается значение 1.

Следовательно, получена двоичная дробь 0.011.

Следует отметить, что не всегда путем повторения операций умножения можно достичь результата умножения, точно равного 1. В таком случае процесс повторения останавливают по достижении необходимой точности, а целую часть результата последней операции умножения используют в качестве значения младшего значащего разряда.

Рассмотрим еще пример преобразования десятичной дроби в двоичную.

Пример. Преобразование десятичного числа $0{,}3_{10}$ в двоичное:

Умножение	Результат в целочисленной форме
$2 \times 0{,}3 = 0{,}6$	0
$2 \times 0{,}6 = 1{,}2$	1
$2 \times 0{,}2 = 0{,}4$	0
$2 \times 0{,}4 = 0{,}8$	0
$2 \times 0{,}8 = 1{,}6$	1
$2 \times 0{,}6 = 1{,}2$	1
$2 \times 0{,}2 = 0{,}4$	0
$2 \times 0{,}4 = 0{,}8$	0
$2 \times 0{,}8 = 1{,}6$	1
$2 \times 0{,}6 = 1{,}2$	1
$2 \times 0{,}2 = 0{,}4$	0

Здесь процедура преобразование носит характер бесконечного построения группы одинаковых операций и результатов. Поэтому ограничимся восемью разрядами. Тогда получим $0{,}3_{10} = 0{,}01001100_2$.

Шестнадцатеричная и восьмеричная системы счисления. Преобразования из одной системы в другую.

В шестнадцатеричной системе счисления используются следующие 16 символов: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E и F. Практическое использование шестнадцатеричной системы объясняется тем, что число 16 есть число 2 в четвёртой степени. Иначе говоря, шестнадцатеричную цифру можно использовать как средство сокращенной записи 4-разрядного двоичного числа (тетрады). В табл.1.2 приводятся примеры шестнадцатеричных чисел и их двоичных и десятичных эквивалентов.

Таблица1.2. Шестнадцатеричные числа и их двоичные и десятичные эквиваленты

Шестнадцатеричное число	Двоичное число	Десятичное число
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Процедура преобразования двоичного числа в шестнадцатеричное довольно проста. Биты, начиная с младшего значащего бита (расположенного рядом с двоичной точкой), объединяются в группы по четыре. Каждой группе подбирается соответствующий шестнадцатеричный символ.

Рассмотрим два примера. Чтобы преобразовать двоичное число 10101011111101_2, необходимо добавить слева два незначащих нуля с целью формирования битов в группы по четыре: 0010 1010 1111 1101. Заменив каждую группу битов соответствующим шестнадцатеричным символом, получим число $2AFD_{16}$. В результате преобразования двоичного числа 11000 111_2 в шестнадцатеричное получим число $C7_{16}$. Сравним исходные данные и результаты обоих примеров, нетрудно заметить, что шестнадцатеричная форма записи числа много проще и легче воспринимается, чем двоичная.

Преобразование двоичных дробей в шестнадцатеричные осуществляется по правилам, аналогичным для преобразования этих дробей в восьмеричные. Для этой цели биты дробной части, начиная со старшего значащего бита

(расположенного справа от двоичной точки), группируются по четыре. Добавление незначащих нулей осуществляется по мере необходимости. Например, для преобразование двоичной дроби $0{,}0101101_2$ в шестнадцатеричную биты группируют по четыре: 0,0101 1010. Затем, каждую группу заменяют шестнадцатеричным символом, получая в результате $0{,}5A_{16}$. Подобным образом, преобразуя число 1101.0111_2, составляют группы 1101.0111, и после их замены шестнадцатеричными символами формируется число $D.7_1$.

Восьмеричная запись, как и шестнадцатеричная, используется для представления двоичных чисел. Восьмеричная система содержит 8 цифр от 0 до 7 и является соответственно системой с основанием 8. В табл. 6.8 представлено несколько десятичных, восьмеричных и двоичных чисел.

Преобразуем двоичное число 11111000100 в его восьмеричный эквивалент. Процедура действий в этом случае следующая. Начиная с МБ двоичного числа, делим его на группы из 3 бит. Затем, используя табл.1.3, преобразуем каждую триаду (группу из 3 бит) в эквивалентную восьмеричную цифру. Таким образом, мы заменим двоичное число 11111000100 его восьмеричным эквивалентом 3704_8:

Двоичное число 011111000 100

Восьмеричное число 3704

Таблица 1.3. Десятичные, восьмеричные и двоичные эквиваленты

Десятичные	Восьмеричные	Двоичные		
		4	2	1
0	0	0	0	0
1	1	0	0	1
2	2	0	1	0
3	3	0	1	1
4	4	1	0	0
5	5	1	0	1
6	6	1	1	0
7	7	1	1	I

Преобразуем теперь восьмеричное число 6521 в его двоичный эквивалент. Каждая восьмеричная цифра заменяется двоичной триадой и получится, что $6521_8 = 110101010001_2$:

Восьмеричное число 6 5 2 1

Двоичное число 110 101010001

Преобразование восьмеричных или шестнадцатеричных чисел в десятичные схоже с преобразованием этих чисел в их двоичные эквиваленты. Каждой позиции числа присваивается определённый вес. Затем значение веса позиции умножается на цифру, занимающую эту позицию. Результаты операции умножения, выполненных для всех позиций числа, суммируются. Следующие ниже примеры демонстрируют преобразование чисел из восьмеричной и

шестнадцатеричной систем в десятичную.

Пример 1. Записать восьмеричное число 2357 в десятичной форме.

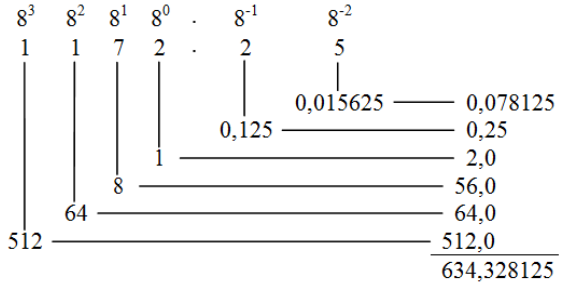
Классическая процедура выполняется согласно табл.1.4. Здесь 512, 64, 8 и 1 есть веса четырех первых восьмеричных позиций. В этом примере содержится 7 единиц, 5 восьмерок, 3 числа 64 и два числа 512. Мы их складываем и получаем результат:

$$1024 + 192 + 40 + 7 = 1263_{10}.$$

Таблица 1.4. *Восьмерично-десятичное преобразование*

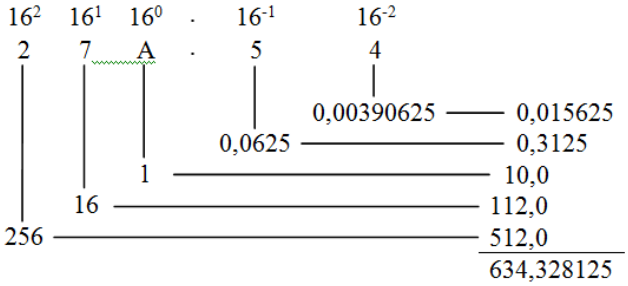
Степень восьми	8^4		8^2		8^1		8^0	
Значеня позицій	512		64		8		1	
Восьмеричное число	2 512 x 2		3 64 x 3		5 8 x 5		7 1 x 7	
Десятичное число	1024	+	192	+	40	+	7	$= 1263_{10}$

Пример 2. Представить в десятичной системе восьмеричное число 1172.25_8:



Результат: $1172.25_8 = 634.328125_{10}$.

Пример 3. Представить в десятичной системе шестнадцатеричное число $27A.54_{16}$:



Результат: $27A.54_{16} = 634.328125_{10}$.

Описанная процедура в действительности весьма проста. Что же касается дробной части, то на примере двоичных дробей можно было убедиться, что число может оказаться весьма длинным. Поэтому часто приходиться прибегать к округлению.

Рассмотрим теперь преобразование десятичных чисел в восьмеричные и шестнадцатеричные. Можно воспользоваться процедурой, подобной той, с

помощью которой осуществлялось преобразование десятичных чисел в двоичные, поскольку она в определённом смысле универсальна. Для преобразования десятичного числа в восьмеричное или шестнадцатеричное вместо деления (умножения для дробей) исходного числа на 2 необходимо выполнить деление (умножение) на 8 или 16. Остатки (целые части произведений) используются для формирования результата.

Ниже приводятся два примера преобразования десятичного числа в восьмеричное и шестнадцатеричное. Процедуры обработки целой и дробной частей десятичного числа отличаются друг от друга и выполняются порознь. Результаты объединятся при формировании восьмеричного или шестнадцатеричного эквивалентов.

Пример 1. Преобразовать десятичное число 634.328125_{10} в восьмеричное:

Преобразование целой части разование дробной части

Преобразование целой части				Преобразование дробной части	
Шаг	Деление	Частное	Остаток	Умножение	Произведение (целая часть)
1	634/8	79	2 (МЗР)	$8 \times 0{,}328125 = 2{,}625$	2 (СЭР)
2	79/8	9	7	$8 \times 0{,}625 = 5{,}0$	5 (МЗР)
3	9/8	1	1		
4	1/8	0	1 (СЭР)		

Результат: $634_{10} = 1172_8$ $0{,}328125_{10} = 0{,}25_8$

Общий результат: $634{,}328125_{10} = 1172{,}25_8$

Пример 2. Преобразовать десятичное число 634.328125_{10} в шестнадцатеричное:

Преобразование целой части				Преобразование дробной части	
Шаг	Деление	Частное	Остаток	Умножение	Произведение (целая часть)
1	634/16	39	$10_{10} = A_{16}$ (МЗР)	$16 \times 0{,}328125 = 5{,}25$	5 (СЗР)
2	39/16	2	7	$16 \times 0{,}25 = 4{,}0$	4 (МЗР)
3	2/16	0	2 (СЗР)		

Результат: $634_{10} = 27A_{16}$ $0{,}328125_{10} = 0{,}54_{16}$

Общий результат: $634{,}328125_{10} = 27A,\ 54_{16}$

Преобразование десятичных чисел в их восьмеричные или шестнадцатеричные эквиваленты выполняется с целью получения более компактной формы представления исходных десятичных данных.

Существуют также процедуры преобразования восьмеричных чисел в шестнадцатеричные, и наоборот. Однако необходимость в их использовании возникает крайне редко. Но если такая проблема появилась, одним из способов её решения является представления исходного числа в двоичной форме, а затем преобразование полученного двоичного числа в требуемую систему счисления.

Задания для практической работы

1. Преобразовать в десятичный код следующие двоичные числа:
   а) 0001; б) 0101; в) 1000; г) 1011; д) 1111; е) 0111.
2. Преобразовать в десятичный код следующие двоичные числа:
   а) 1000 0000; б) 0001 0000; в) 0011 0011; г) 0110 0100; д) 0001 1111.
3. Преобразовать в двоичный код следующие десятичные числа:
   а) 23; б) 39; в) 55; г) 48.
4. Записать следующие шестнадцатеричные числа в двоичной форме:
   а) C; б) 6; в) F; г) E; д) 1A; е) 3D; ж) A0; з) 8B; и) 45; к) D7.
5. Преобразовать следующие двоичные числа в шестнадцатеричный код:
   а) 1001; б) 1100; в) 1101; г) 1111; д) 1000 0000; е) 0111 1110; ж) 001 0101.
6. Преобразовать следующие шестнадцатеричные числа в десятичный код:
   а) 7E; б) DB; в) 12A3; г) 34CF.
7. Записать следующие восьмеричные числа в двоичном коде:
   а) 3; б) 7; в) 0; г) 7642; д) 1036; е) 2105.
8. Записать следующие двоичные числа в восьмеричном коде:
   а)101; б)110; в)010; г)111000101010; д)1011000111; е) 100110100101.
9. Записать восьмеричное число 2357 в десятичной форме.
10. Записать десятеричное число 2648_{10} в 8-ричном коде.
11. Преобразовать десятичное число 0.34375_{10} в двоичное.

Содержание отчёта

Отчет должен содержать название и цель работы, алгоритм преобразования чисел из одной системы счисления в другую и решения домашнего задания.

Практичні заняття № 2

ДВІЙКОВА АРИФМЕТИКА ТА АРИФМЕТИКА У ДОДАТКОВОМУ КОДІ ОПЕРАНДІВ

Цель практических занятий – изучить алгоритм действий при решении задач двоичной арифметики и арифметики в двоичном коде.

В отличии от людей, у которых есть различные арифметические операции (сложение, деление, умножение…), в микропроцессорах присутствует только одна арифметическая операция – сложение, с помощью которой они производят вычисления различных алгоритмов. Например, операция 7–3 для микропроцессора будет выполняться в виде 7 + (-3), где 7 и -3 представлены в двоичной системе.

Двоичное сложение в микропроцессоре не слишком сильно отличается от операции сложения в столбик для десятичной системы счисления. Отличие состоит лишь в том, что в десятичной системе счисления при достижении 10

происходило присвоение текущему разряду 0 и добавления 1 к следующему, а для двоичной системы это будет происходить при достижении 2.

На рис. 2.1,а представлены простые правила двоичного сложения. Два первых (слева) правила очевидны, третье показывает, что 1 + 1 = 10, т. е. наиболее значимая 1 переносится в ближайший старший разряд. Четвертое правило показывает, что 1 + 1 + 1 = 11. В этом случае первое, второе слагаемые и запоминаемое в результате сложения в младшем разряде число — все 1. Результатом является сумма — 1 с переносом 1.

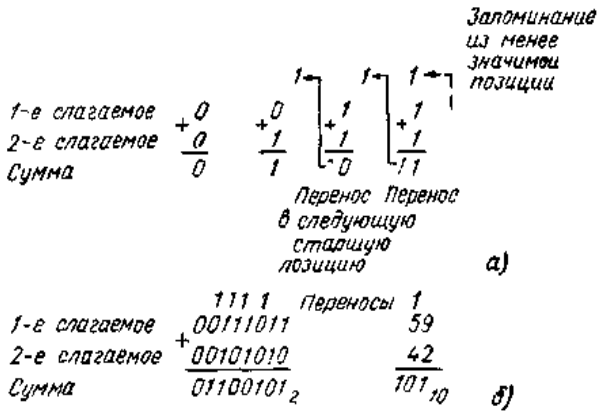


Рис. 2,1. Двоичное сложение: а — правила: б — пример

Сложим двоичные числа 0011 1011 и 0010 1010 (операция показана на рис. 2.1,б). Для большей ясности действия с десятичными эквивалентами обрабатываемых чисел показаны на рисунке справа. Суммой двух чисел 0011 1011 и 0010 1010 будет 0110 0101_2.

Сложение начинается с крайнего правого разряда. Две единицы младшего разряда объединяются в единицу старшего.

Пример: $1011{,}1_2 + 1010{,}11_2$

```
  1←  1← 1← 1←
    1 0 1 1,1 0
+   1 0 1 0,1 1
-------------------
1 0 1 1 0,0 1
```

На рис. 2.2,а приведены правила двоичного вычитания. Первые три аналогичны десятичному вычитанию. Последнее требует заема из более значимого предшествующего разряда (в этом случае вес 2). Уменьшаемым является двоичное число 10, вычитаемым 1, разностью—1.

Вычтем двоичное число 0011 1001 из 0101 0101. Этот пример приведен на рис. 2.2,б. Разряды весов 1, 2 и 4 этого двоичного вычитания просты для выполнения и относятся к первым трем правилам на рис. 2.2,а. В колонке веса 8 имеет место вычитание 1 из 0. Тогда 1 занимается из колонки веса 16. Единица вычитается из 10_2, что дает разность 1 согласно четвертому правилу на рис. 2.2,а. После этого заема в колонке веса 16 имеет место вычитание 1 из нового вычитаемого 0. Согласно четвертому правилу 1 должна быть занята из

следующей, более значимой позиции (колонка веса 32), но в колонке 32 имеем 0; поэтому колонка 32 должна сделать заем из колонки веса 64, что и вы полнено. Окончательно колонка 16 делает заем из колонки 32, уменьшаемым в колонке 16 становится 10_2, вычитаемым 1, разностью 1. В колонке 32 имеем 1 - 1 = 0, в колонке 64 — 0 - 0 = 0, в колонке 128 — 0 - 0 = 0. Таким образом, рис. 2.2, б иллюстрирует операцию вычитания 0011 1001_2цз 0101 0101_2 (справа эта задача решена в десятичной записи) .

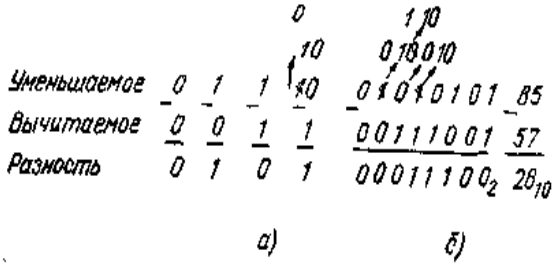


Рис. 2.2. Двоичное вычитание: *а* — правила; *б* — пример

Таким образом, при вычитании (0 - 1) в разряде разности получается 1, разряды уменьшаемого, начиная со следующего, изменяются на противоположные (инвертируются) до первой встречной единицы (включительно). После этого производится вычитание из измененных разрядов уменьшаемого.

Рассмотрим ещё несколько примеров вычитания многоразрядных чисел (из большего числа вычитается меньшее).

```
 0 1 1 1      ← изменение уменьшаемого в результате займа
_1 0 0 0 0 1 = 33
   0 1 0 1 1 = 11
-------------
 0 1 0 1 1 0 ← разность=22

 0 1 0 1 1    ← изменение уменьшаемого в результате займа
_1 0 1 0 0 0 = 40
   1 1 0 1 1 = 27
-------------
 0 0 1 1 0 1 ← разность = 13
```

Очевидно, что как в десятичном, так и в двоичном коде, складывать значительно проще, чем вычитать. Поэтому большое распространение получила двоичная арифметика с учетом знаков чисел, где вычитание заменяется сложением чисел с учетом их знака. При этом уже не имеет значения соотношение чисел между собой, какое из них больше - вычитаемое или уменьшаемое. Знак разности получается автоматически.

В двоичном коде знак числа представляет собой разряд, приписываемый слева от значащих разрядов числа. Знак " + " обозначается логическим 0, знак " – " - логической 1.

Микропроцессор обрабатывает информацию обычно в двоичном коде. Однако если нужно использовать числа со знаком, используется специальный дополнительный код. (*Далее все примеры будем рассматривать для целых чисел, отделяя знаковый разряд точкой.)*

Прямой код (ПК) для отрицательных и для положительных чисел образуется одинаково, простым дописыванием знакового разряда.

Так, в восьмиразрядном формате

Обратный код (ОК) для положительных чисел совпадает с прямым, т.е. к значащим разрядам приписывается знаковый разряд. Для отрицательных чисел значащие разряды инвертируются (нули заменяются на единицы, единицы - на нули), после чего приписывается знак.

Для того же числа *обратный код* имеет вид:

$$(+10) = 0.0001010,\ (-10) = 1.1110101.$$

Дополнительный код (ДК) для положительных чисел совпадает с обратным и прямым, т.е. к значащим разрядам приписывается знаковый разряд. Для отрицательных чисел дополнительный код на 1 больше, чем обратный. После образования значащих разрядов приписывается знаковый разряд.

Для значащих разрядов отрицательного числа справедлива формула:

ДК=ОК+1

Рассмотрим образование дополнительного кода для числа 10. Для положительного числа $(+10) = 0.0001010$, а для отрицательного числа (-10) дополнительный код получается следующим образом:

$$\begin{array}{rr} & 0\ 0\ 0\ 1\ 0\ 1\ 0 \leftarrow \text{ПК} \\ + & 1\ 1\ 1\ 0\ 1\ 0\ 1 \leftarrow \text{ОК} \\ & \qquad\qquad\ 1 \\ \hline \text{приписываем знак числа} \rightarrow 1. & 1\ 1\ 1\ 0\ 1\ 1\ 0 \leftarrow \text{ДК} \end{array}$$

Для замены вычитания сложением применяется и обратный, и дополнительный коды, при этом в каждом из них действуют свои правила.

Двоичная арифметика в дополнительном коде

При решении рекомендуется придерживаться следующей последовательности действий. Допустим, заданы два десятичных числа, тогда необходимо:

1. перевести десятичные числа в двоичный код;
2. уравнять форматы полученных двоичных чисел;
3. если знаки чисел одинаковые, добавить по одному резервному нулю слева от каждого числа во избежание переполнения;
4. получить дополнительные коды чисел;
5. приписать знаковые разряды;
6. сложить полученные коды по правилам двоичного сложения;
7. перенос из знакового разряда (если он есть) отбросить;

8. результат получен в дополнительном коде, поэтому для проверки значащих разрядов отрицательного числа необходимо сначала вычислить обратный код по формуле $\text{ОК} = \text{ДК} - 1$, после чего произвести инверсию полученного числа.

Для наглядности возьмем два десятичных числа, например, 20 и 55, и сделаем все возможные варианты вычислений:

1) $$20 + 55 = (+20) + (+55) = +75.$$

$$\begin{array}{l} \downarrow \\ \quad 1\ 1\quad 1 \qquad \leftarrow \text{перенос} \\ +0.\ 0\ 0\ 1\ 0\ 1\ 0\ 0 = +20 \\ \quad 0.\ 0\ 1\ 1\ 0\ 1\ 1\ 1 = +55 \\ \hline \quad 0.\ 1\ 0\ 0\ 1\ 0\ 1\ 1 \leftarrow \text{результат} \\ \quad \uparrow \end{array}$$

резервный разряд во избежание переполнения — знаковый разряд положительного числа

Число положительное, поэтому ОК=ПК, для проверки числа нужно перевести его значащие разряды в десятичный код:

$$1001011_2 = 1 + 2 + 8 + 64 = 75.$$

2) $$20 - 55 = (+20) + (-55) = -35.$$

Сначала получим дополнительный код отрицательного числа (-55):

$$\begin{array}{r} 1\ 1\ 0\ 1\ 1\ 1 \leftarrow \text{ПК} \\ +0\ 0\ 1\ 0\ 0\ 0 \leftarrow \text{ОК} \\ \hline 1 \\ \hline \text{приписываем знак числа} \rightarrow\ 1.\quad 0\ 0\ 1\ 0\ 0\ 1 \leftarrow \text{ДК} \end{array}$$

Здесь важно уяснить, что крайние левые нули в значащих разрядах сокращать нельзя, поскольку они являются значимыми. Иными словами, все вычисления для каждого примера производятся в неизменном формате, в данном случае в примере (б) - это шесть значащих разрядов, т.е. столько, сколько содержится в большем числе.

$$\begin{array}{l} +0.\ 0\ 1\ 0\ 1\ 0\ 0 = +20 \\ \quad 1.\ 0\ 0\ 1\ 0\ 0\ 1 = -55 \\ \hline \quad 1.\ 0\ 1\ 1\ 1\ 0\ 1 \leftarrow \text{результат} \\ \quad \uparrow \end{array}$$

знаковый разряд отрицательного числа

Вновь получили знак числа и его значащие разряды, занимающие жестко заданные позиции в выбранном формате числа. Поскольку получено отрицательное число, то ДК ПК, для проверки его значащих разрядов нужно сначала вычислить *обратный код*, затем перевести его в прямой код инверсией –

$$\begin{array}{r} -0\ 1\ 1\ 1\ 0\ 1 \leftarrow \text{ДК} \\ \hline 1 \\ \hline 0\ 1\ 1\ 1\ 0\ 0 \leftarrow \text{ОК} \\ \hline 1\ 0\ 0\ 0\ 1\ 1 \leftarrow \text{ПК}, \end{array}$$

а затем уже перевести его в десятичный код:

$$100011_2 = 1 + 2 + 32 = 35.$$

3) $$-20 + 55 = (-20) + (+55) = +35$$ -

Сначала получим ДК отрицательного числа (-20).

$$
\begin{array}{rl}
 & 0\ 1\ 0\ 1\ 0\ 0 \leftarrow \text{ПК} \\
 & \quad 1\ 1 \quad \leftarrow \text{перенос при вычислении ДК} \\
+ & 1\ 0\ 1\ 0\ 1\ 1 \leftarrow \text{ОК} \\
 & \quad\quad\quad 1 \\
\hline
\text{приписываем знак числа} \rightarrow & 1.\ 1\ 0\ 1\ 1\ 0\ 0 \leftarrow \text{ДК}
\end{array}
$$

После этого произведем вычисления:

$$
\begin{array}{rl}
 & \quad\ \downarrow \\
\text{перенос из знакового разряда (отбрасывается)} \rightarrow & \boxed{1}\ 1\ 1\ 1\ 1 \quad\quad \leftarrow \text{перенос} \\
+ & 1.\ 1\ 0\ 1\ 1\ 0\ 0 = -20 \\
 & 0.\ 1\ 1\ 0\ 1\ 1\ 1 = +55 \\
\hline
 & 0.\ 1\ 0\ 0\ 0\ 1\ 1 \leftarrow \text{результат} \\
 & \quad\ \uparrow \\
 & \text{знаковый разряд положительного числа}
\end{array}
$$

Получено положительное число, поэтому $\text{ДК} = \text{ПК}$, для проверки результата нужно только перевести значащие разряды в десятичный код:

$$100011_2 = 1 + 2 + 32 = 35.$$

4) $-20 - 55 = (-20) + (-55) = -75$ -

Сначала получим дополнительный код отрицательных чисел. Для числа (-20) он получается следующим образом:

$$
\begin{array}{rl}
 & \quad\ \downarrow \\
 & \boxed{0}\ 0\ 1\ 0\ 1\ 0\ 0 \leftarrow \text{ПК} \\
\text{резервный разряд во избежание переполнения} \rightarrow & \square \quad\quad 1\ 1 \quad \leftarrow \text{перенос при вычислении ДК} \\
+ & \boxed{1}\ 1\ 0\ 1\ 0\ 1\ 1 \leftarrow \text{ОК} \\
 & \square \quad\quad\quad\quad 1 \\
\hline
\text{приписываем знак числа} \rightarrow & 1.\ \boxed{1}\ 1\ 0\ 1\ 1\ 0\ 0 \leftarrow \text{ДК}
\end{array}
$$

А для числа (-55):

$$
\begin{array}{rl}
 & \quad\ \downarrow \\
 & \boxed{0}\ 1\ 1\ 0\ 1\ 1\ 1 \leftarrow \text{ПК} \\
\text{резервный разряд во избежание переполнения} \rightarrow & \boxed{1}\ 0\ 0\ 1\ 0\ 0\ 0 \leftarrow \text{ОК} \\
 & \square \quad\quad\quad\quad 1 \\
\hline
\text{приписываем знак числа} \rightarrow & 1.\ \boxed{1}\ 0\ 0\ 1\ 0\ 0\ 1 \leftarrow \text{ДК}
\end{array}
$$

Сложим полученные числа в том же формате:

$$
\begin{array}{rl}
\text{перенос из знака (игнорируется)} \rightarrow & \boxed{1}\ 1 \quad\quad 1 \quad\quad\quad \leftarrow \text{перенос} \\
 & 1.\ 1\ 1\ 0\ 1\ 1\ 0\ 0 = -20 \\
+ & 1.\ 1\ 0\ 0\ 1\ 0\ 0\ 1 = -55 \\
\hline
 & 1.\ 0\ 1\ 1\ 0\ 1\ 0\ 1 \leftarrow \text{результат} \\
 & \quad\ \uparrow \\
 & \text{знаковый разряд отрицательного числа}
\end{array}
$$

Поскольку число отрицательное, $\text{ДК} \neq \text{ПК}$. Для проверки значащих разрядов числа нужно сначала вычислить *обратный код*, после чего перевести его в прямой код инверсией -

$$
\begin{array}{rl}
 & \_0\ 1\ 1\ 0\ 1\ 0\ 1 \leftarrow \text{ДК} \\
 & \quad\quad\quad\quad 1 \\
\hline
 & 0\ 1\ 1\ 0\ 1\ 0\ 0 \leftarrow \text{ОК} \\
\hline
 & 1\ 0\ 0\ 1\ 0\ 1\ 1 \leftarrow \text{ПК}.
\end{array}
$$

И только после этого полученное число проверяется переводом в десятичный код:

$$1001011_2 = 1 + 2 + 8 + 64 = 75.$$

Примеры вычисления разности.

Вычисление разности

Десятичная арифметика	Двоичная арифметика	
58	0011 1010	Число 58_{10}
—	+	
23	1110 1001	Дополнительный код числа 23_{10}
35	**1** 0010 0011	Разность 35_{10}

Единица переноса, отбрасываемая в случае положительного результата.

Десятичная арифметика	Двоичная арифметика	
26	0001 1010	Число 26_{10}
—	+	
34	1101 1110	Дополнительный код числа 34_{10}
-08	1111 1000	Разность в форме дополнения (поскольку в старшем разряде 1

Определение абсолютного значения разности

1111 1000	Дополнительный код разности
0000 0111	Обратный код
0000 0001	Единица, добавляемая к обратному коду
0000 1000	Абсолютное значение разности (8_{10})

Умножение и деление двоичных чисел практически не отличается от умножения и деления чисел, записанных в десятичной системе счисления. Единственным отличием является то, что при умножении в столбик не приходится находить произведение первого множителя на значения последовательных разрядов второго множителя, так как значение этих разрядов 1 или 0. А при делении в столбик не нужно подбирать неполное делимое, так как учитывая специфику двоичных чисел, неполное делимое можно определить просто посмотрев на делимое.

Выполнить умножение и деление двоичных чисел, рис.2.3:

1101111 · 101101 = ? 111100 : 1010 = ?

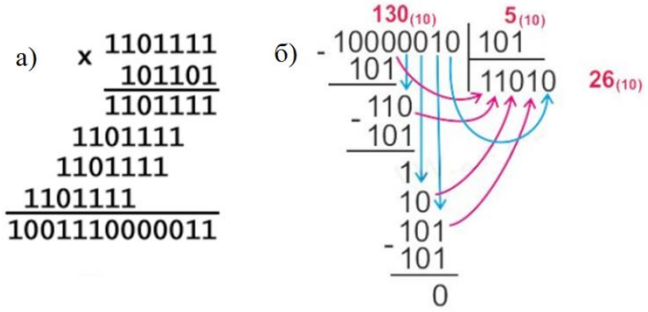


Рис. 2.3 Умножение и деление двоичных чисел

Задания для практической работы.

1. Выполнить следующие сложения двоичных чисел:

а) 1010	б) 1101	в) 0101 1011	г) 0011 1111
+	+	+	+
0101	0101	0000 1111	0001 1111

2. Выполнить следующие вычитания двоичных чисел:

а) 1110	б) 1010	в) 0110 0110	г) 0111 1000
-	-	-	-
1000	0101	0001 1010	00111111,

3. Установить, являются ли следующие числа в дополнительном коде положительными или отрицательными:
а) 0111 0000; б) 1100 1111; в) 1000 1111; г) 0101 0101.

4. Используя табл. 2.10, дать дополнительный код следующих десятичных чисел со знаком:
а) +1; б) +5; в) +127; г) -1; д) -2; е) -128.

5. Используя процедуру, приведенную в § 2.6, дать дополнительный код следующих десятичных чисел со знаком:
а) -10; б) -21; в) -34; г) -96.

6. Используя процедуру преобразования, дать десятичные эквиваленты следующих чисел в дополнительном коде:
а) 1111 1011; б) 0000 1111; в) 1000 1111; г)0111 0111.

7. Сложить следующие десятичные числа со знаком, используя метод дополнительного кода

а) (+7)	б) (+31)	в) (+8)	г) (+89)	д) (+1)	е) (+20)	ж) (—3)
+	+	+	+	+	+	+
(+1)	(+26).	(—5)	(—46).	(—6)	(—60).	(—4)

8. Вычесть следующие десятичные числа со знаком, используя метод дополнительного кода:

а) (+7)	б) (+113)	в) (+3)	г) (+12)
—	—	—	—
(+2)	(+50).	(+8)	(+63).

Содержание отчёта

Отчет должен содержать название и цель работы, алгоритм действий при решении задач двоичной арифметики и арифметики в дополнительном коде, а также решения домашнего задания.

Практичні заняття № 3

ДОСЛІДЖЕННЯ СПОСОБІВ АДРЕСАЦІЇ ОПЕРАНДІВ

/команди переміщення даннх/

Цель работы - изучить способы адресации операндов, составить простейшие программы, исследовать выполнение команд перемещения данных и простых программ.

СГЮСОБЫ АДРЕСАЦИИ

Адрес ячейки памяти или адрес регистра, с которым оперирует команда, можно указать многими способами. В каждом конкретном случае способ адресации выбирается исходя из следующих соображений.

1. Использование команды с возможно более коротким адресом. Эти команды требуют меньшего объема памяти и имеют меньшее время выборки.

2. Обеспечение простого доступа х возможно большему объему памяти. Это противоречит требованию предыдущего пункта. Однако многие программы используют сначала одну область памяти, а затем другую. Тогда появляется возможность легко определять любую часть памяти, используя укороченные адреса, осуществлять доступ к отдельным ячейкам этой части

3. Возможность изменения содержимого адресной части без изменения команды. Одна и та же последовательность команды может быть использована с цепью. обработки всех элементов массива, таблицы нам строки.

4. Обеспечение наиболее выстрой адресации. Предпочтительнее тот способ, который требует меньшего числа операций или дополнительных обращений к памяти .

5. Использование наиболее простых способов адресации. Использование более сложных способов адресации приводит к появлению ошибок в программе.

Наиболее используемыми являются следующие способы адресации.

I. Непосредственная адресация - в адресном попе записывается само данное.

2. Прямая адресация - в адресном поле записывается истинный адрес операндов /ячейка памяти/.

3. Регистровая адресация - в адресном поле записывается регистр.

4. Косвенная регистровая адресация - в адресном поле записывается пара регистров *HL*, которая обозначается буквой *M*.

НЕПОСРЕДСТВЕНАЯ АДРЕСАЦИЯ

При этом методе задания операнда данные, подлежащие обработке микропроцессором, содержатся непосредственно в самой команде в виде ее второго слова. Таким образом, никакого адреса не требуется, поэтому вместо "непосредственной адресации" довольно часто говорят "непосредственное

представление". Мнемоническое обозначение команды – *MVIr*, что означает загрузку регистра вторым байтом команды. Символическое наименование регистров */ r =A , B , C , D , E , H , L /* следует сразу за обозначением команды. Например, *MVIB*, 12, т.е. по этой команде число 12 заносится в регистр *B*.

Для загрузки в память машины все команды должны быть преобразованы в объектное поле кода с помощью шестнадцатеричной системы кодирования. Для облегчения этой процедуры существует специальная таблице, строки которой представляют собой старшие разряды кода операций, а столбцы - младшие разряды. На пересечении указаны мнемокоды команд, соответствующих образовавшихся комбинациям.

Ассемблерные программы записываются в виде последовательности команд, называемых также операторами.

Предположим, необходимо составить программу непосредственной загрузки регистра-аккумулятора числом *OA*. Она будет иметь вид :

Адрес	ашинный код	Мнемоника команды	Комментарий
№ № 00	*3E*	*M VIA, OA*	Загрузить в
№ № 01	*0A*		акумулятор число *OA*
№ № 02	*76*	*HLT*	Останов

Для загрузки начальных значений во внутренние указатели памяти /инициализации/ применяют трехбайтовую команду с непосредственным представлением *LXIrp*, где *p* означает регистровые пары *BC,DE, HL*или указатель стена *SP*.

Например, необходимо загрузить регистровую пару *HL*, адресом *№№ 00*. Соответствующая этому действию программа выглядит так:

№№ 00	*2I*	*LXI H,NNOO*	Загрузить регистровую
№№ 01	*00*		HL пару адресом
№№ 02	*№№*		*№№ 00*
№№ 03	*76*	*HTL*	Останов

ПРЯМАЯ ААРЕСАЦИЯ

При такой адресации команды имеют трехбайтовый формат. Первый предназначен для кода операции, второй и, если имеется, третий - для адреса. Адрес указывает область, в которой находятся подлежащие обработке данные. Таким образом, при прямой адресации возможно в случае запоминания поместить содержимое аккумулятора в ячейку памяти, адрес которой указан вторым и третьим байтом., а при загрузке, наоборот, поместить в аккумулятор данные из ячейки памяти, адресуемой аналогично.

При необходимости загрузки аккумулятора и запоминания данных используются команды:

LDA - загрузить

STA - запомнить

Пример. Число *2A*, записанное в ячейку *№№ 00* , извлечь и записать по адресу *№№ I0*.

№№ 00	*2A*		Записать число *2A* по адресу *№№ 00*
№№ 01	*3A*	*LDA NNOO*	Загрузить аккумулятор числом, находящимся по адресу *№№ 00*
№№ 02		*00*	
№№ 03		*№№*	
№№ 04	*32*	*STA NNIO*	Число в аккумуляторе записывается по адресу *№№ I0*
№№ 05		*I0*	
№№ 06		*№№*	
№№ 07	*76*	*HLT*	Останов

Для загрузки основного указателя памяти регистров *H* и *L* из двух смежных ячеек памяти применяется. команда *LHLD* - загрузить в пару регистров *HL* .

Запоминание содержимого регистров *H* и *L* в двух следующих друг за другом ячейках памяти осуществляется по команде *SHLD* - запомнить содержимое пары регистров *HL*.

Обе эти команды трехбайтовые, во-втором и третьем байте указывается полный I6-разридный адрес ячейки, из которой загружается или в которую заносится содержимое регистра *L* /младшего байта/. Адрес содержимого регистра *H* /старшего байта/ получается прибавлением *I*к значению второго и третьего байта команды.

Пример. Число *3CO3*, записанное в ячейки памяти *№№ 00* и *№№ 01*,загрузить в регистровую пару *HL*.

№№ 00	*03*		Число *03* записать в ячейку *№№ 00*
№№ 01	*3C*		Число *3C* записать в ячейку *№№ 01*
№№ 02	*2A*	*LHLD NNOO*	Загрузить в пару регистров *HL* число ,
№№ 03		*00*	занесенное в ячейки *№№ 00* и *№№ 01*
№№ 04		*№№*	
№№ 05	*76*	*HTL*	Останов

В группу команд прямой адресации включаются также двухбайтовые команды ввода*IN* и вывода *OUT* . Второй байт этих команд представляет собой 8-битный адрес порта ввода /вывода/, а получателем /источником/ может быть только аккумулятор. Таким образом, команда *IN*идентична команде *LDA*-загрузить.

Примером использования команд *IN*и *OUT* является следующая программа.

№№ 00	*DA*	*IN20*	Загрузить аккумулятор числом со входного
№№ 01	*20*		порта *20*
№№ 02	*D3*	*OUT 30*	Записать число в выходное устройство *30*

№№ 03	*30*		
№№ 04	*76*	*HTL*	Останов

РЕГИСТРОВАЯ АДРЕСАЦИЯ

Основной командой является однобайтовая команда перемещения данных из одного регистра в другой. В зоне операнда должны быть указаны два адреса: на первом месте - куда помещается информация, т.е. адреса получателя /приемника/; на втором месте - откуда берутся данные, т.е.адрес источника.

При операции перемещения данных содержимое источника сохраняется неизменным, а начальное содержимое приемника заменяется результатом операции.

В мнемонике команды *MDV*r_1,r_2 указываются условные обозначения регистров общего назначения - приемника и истопника. Между собой эти обозначения обязательно резделяются запятой. Например, *MDVC, B* означает перенести содержимое регистра *B* в регистр *C*; содержимое регистра *B* остается без изменения. Команда перемещения данных имеет .самое большое число возможных вариантов. Если перебрать все возможные комбинации регистров, то в перечне машинных команд будет занято 64 позиции.

Примером регистрового способа адресации служит программа:

№№ 00	*06*	*MVIB , 10*	Загрузить регистр *B* числом *I0*
№№ 01	*I0*		
№№ 02	*48*	*MOV C , B*	Передать содержимое регистра *B* в регистр *C*
№№ 03	*76*	*HTL*	Останов

Особенностью микропроцессора являются однобайтовые команды передач I6-битных операторов. Команды, SP*HL и PCHL* передают содержимое регистров *HL*, в указатель стека и программный счетчик *PC* соответственно. Если рассмотреть внимательно мнемонику команды *SPHL* - передачи содержимого пары регистров *HL* в указатель стека *SP* , подтвердится, что приемник указывается первым / *SP* /, а источник – последним / *HL*/, Точно так же и в команде *PCHL*.

Существует команда обмена *XCHG* ,производящая обмен содержимого регистров */DE* / и */HL /*.

Пример. Число *FA01* передать из регистровой пары *HL* в указатель стека *SP*.

№№ 00	*21*	*LXIH FAO1*	Загрузить регистровую пару числом *FA 01*
№№ 01		*01*	
№№ 02		*FA*	
№№ 03	*F9*	*SPHL*	Передать содержимое регистровой пары *HL* в указатель стека *SP*
№№ 04	*76*	*HLT*	Останов

Пример. Содержимое регистров *DE -0152* и *HL –E09A* поменять местами.

№№ 00	*21*	*LXIH , EOSA*	Загрузить регистровую пару*HL*

№№ 01		*9A*	числом*E09A*
№№ 02		*EO*	
№№ 03	*11*	*LXID , 0152*	Загрузить регистровую пару*DE*
№№ 04		*52*	числом*0152*
№№ 05		*01*	
№№ 06	*EB*	*XCHG*	Поменять содержимое регистров *DE*и*HL*
№№ 07	*76*	*HTL*	Останов

КОСВЕННАЯ РЕГИСТРОВАЯ АДРЕСАЦИЯ

Если данные перемещаются в регистр из основной памяти или, наоборот, переносятся в память из регистра, то используется косвенная регистровая адресация. При этом в качестве I6-битного адреса ячейки памяти используются данные, находящиеся в паре регистров *H* и *L*. В составе команд адресации пара регистров *HL* обозначается буквой *M*.

Например,*MDVM, B*означает: поместить содержимое регистра *B* в ячейку памяти, адрес которой находится в регистрах *H* и *L*. Команда *MDV B*, *M* аналогична предыдущей, но перемещение информации происходит в противоположном направлении–из памяти /по адресу в регистрах *H* и *L*/ в регистр-приемник *B*. Предположим, необходимо занести в ячейку памяти, адрес которой указан в регистровой перечень регистр *E*.

№№ 00	*21*	*LXIH , NN20*	Занести в регистровую пару *HL*адрес
№№ 01		*20*	*№№ 20*
№№ 02		*№№*	
№№ 03	*3E*	*MVIA , FF*	Загрузить аккумулятор числом *FF*
№№ 04		*FF*	
№№ 05	*36*	*MVIM , A*	Записать в ячейку памяти , адрес которой указан в*HL* , содержимое аккумулятора *A*
№№ 06	*5E*	*MOVE , M*	Передать содержимое ячейки, память которой указан в *HL* , регистр *E*
№№ 07	*76*	*HTL*	Останов

При необходимости загрузки аккумулятора и запоминания данных в случае косвенной регистровой адресации используются команды:

LDAX – загрузить, *STAX* – запомнить с косвенной адресацией.

Эти команды имеют однобайтовый формат. При запоминании они вызывают запись содержимого аккумулятора в ячейку 03У, 16-битный адрес которой указывается парой регистров. Это может быть не только пара *HL*, но и *BC* либо*DE*. При загрузке адресация делается аналогично, но данные записываются в аккумулятор.

Например, необходимо записать в аккумулятор число *FO* из ячейки памяти, адрес которой указывается парой регистров *DE*.

№№ 00	*F0*		Записать число*F0* в ячейку *№№ 00*
№№ 01	*11*	*LXID NNOO*	Записать в регистровую пару*DE* адрес числа*F0*
№№ 02			
		00	
№№ 03		*№№*	
№№ 04	*1A*	*LDAXD*	Записать в аккумулятор число , адрес которого указывается парой *DE*
№№ 05	*76*	*HLT*	Останов

ЗАДАНИЕ К ПРАКТИЧЕСКОЙ РАБОТЕ

1. Составить 9-10 программ смешанной адресации данных, используя в каждой из них не менее трех различных команд перемещения данных.

СОДЕРЖАНИЕ ОТЧЕТА

Отчет должен содержать название работы, цель, перечень команд адресации данных с комментариями, программы, разработанные и исследованные согласно заданию к практической работе, анализ полученных результатов и выводы.

Практичні заняття №4

РОЗРОБКА ТА ДОСЛІДЖЕННЯ ПРОГРАМ З ВИКОРИСТАННЯМ КОМАНД АРИФМЕТИЧНИХ ДІЙ

Цель работы - изучить способы организации исследования программ выполнения операций сложения и вычитания.

Во всех командах, связанных с преобразованием данных, в качестве подразумеваемого и поэтому не адресуемого операнда используется содержимое важнейшего внутреннего регистра

/А-регистра/. Результат операции обычно также загружается в аккумулятор. Таким образом, аккумулятор является узловой точкой всех преобразования данных и основным рабочим регистром микропроцессора. Он выполнен как регистр-фиксатор, вход которого связан с выходом АЛУ, а выход подключается к одному из входов АЛУ.

Команды арифметических действий содержат команды сложить, вычесть, инкрементировать /увеличить на 1/, сравнить.

Существуют 3 группы команд сложения. Аккумулятор /регистр А/ содержит одно из слагаемых. Каждая команда точно оговаривает различные источники другого слагаемого.

Команда *ADI*- сложить *А* с данными по непосредственной адресации - является двухбайтовой. Код выполняемой операции содержится в первом байте команды, непосредственно за ним содержатся данные для сложения с

содержимым аккумулятора. Данные, находящиеся в памяти непосредственно за кодом операнда /*КОП*/, складываются с содержимым аккумулятора. Сумма помещается в аккумулятор. Так, программа сложения однобайтовых чисел *OA* и *OB* имеет вид:

№№ 00	*3E*	*MVIA, 0A*	Загрузить регистр аккумулятора числом *OA*
№№ 01	*0A*		
№№ 02	*C6*	*ADI OB*	К содержимому аккумулятора прибавить число *OB*
№№ 03	*0B*		
№№ 04	*76*	*HTL*	Останов

Вторая группа команд сложения используется при регистровой адресации *ADDB / C, D,E, H, L, /* - сложить *A* с данными *B, C, D, E, H, L.*

Примером сложения двух чисел *IO + 2I* с использованием команды *ADD* является следующая программа:

№№ 00	*06*	*MVIB, 10*	Загрузить регистр *B* числом *10*
№№ 01	*10*		
№№ 02	*16*	*MVID, 21*	Загрузить регистр *D* числом *21*
№№ 03	*21*		
№№ 04	*76*	*MOVA, B*	Передать содержимое регистра *B* в регистр *A*
№№ 05	*82*	*ADDD*	К содержимому аккумулятора прибавить содержимое регистра *A*
№№ 06	*76*	*HTL*	Останов

В процессе сложения двух и более однобайтовых чисел может оказаться, что результат представляет собой двухбайтовое число. Например, аккумулятор содержит *FF*, что в двоичной системе выражается как $IIIIIII_2$, а регистр *B* содержит число *0I*, которое в двоичной системе выражается как $0000000I_2$. Выполнив операцию сложения, аккумулятор содержит в младших битов суммы, т.е.00000000_2. Индикатор "флаг переноса"*C* микро-ЭВМ регистра состояния установлен в *I,* что показывает наличие переноса старшего бита результата. Индикатор "флаг нуля" *Z* микро-ЭВМ проверяет содержимое аккумулятора после операции сложения – 00000000_2.

Последняя команда сложения представляет собой однобайтовую команду сложить с косвенным регистром /*ADDM*/.

Адрес данного слагаемого числа задан в более сложной форме с использованием способа косвенной регистровой адресации. В случае применения данной команды пара регистров *HL* указывает 16-разрядный адрес памяти, содержимое которой складывается с содержимым аккумулятора, и сумма помещается в аккумулятор. Команды косвенного сложения используют в качестве указателя адреса 16-разрядный регистр /обычно пару *HL* /.

Предположим, необходимо составить программу сложения двух чисел *0F* и *2B*, используя косвенную регистровую адресацию, и результат сложения поместить в ячейку памяти *M* с определенным адресом.

№№ 20	*0F*		Число *0F* помещено в ячейку памяти *№№ 20*
№№ 00	*26*	*MVIH, NN*	Загрузить регистр *H* старшим байтом памяти *№№*
№№ 01		*№№*	
№№ 02	*2E*	*NVIL, 20*	Загрузить регистр *L* младшим байтом памяти *20*
№№ 03		*20*	
№№ 04	*3E*	*MVIA, 2B*	Загрузить аккумулятор числом *2B*
№№ 05		*2B*	
№№ 06	*86*	*ADDM*	Сложить содержимое ячейки памяти с адресом , указанным в регистрах *HL*с данными в аккумуляторе
№№ 07	*77*	*MOV M, A*	Передать содержимое аккумулятора в ячейку памяти б адрес которой определяется содержимым регистров *HL*
№№ 08	*76*	*HTL*	Останов

Часто необходимо просуммировать несколько чисел, занесенных в ячейки памяти с последовательной нумерацией адресов. Для упрощения программ сложения вводится команда *INXrp*, где *rp* означает регистровую пару /*B, D, H*/.

Эта команда позволяет увеличить на единицу /инкрементировать/ содержимое регистровой пары, в которой указывается адрес первого слагаемого.

Для примера составим последовательную программу сложения чисел трех последовательных ячеек памяти с последующим размещением результата суммы в свободной ячейке памяти.

№№ 20	*0F*		Три слагаемых введены в память по адресам *№№ 20 - №№ 21*
№№ 21	*09*		
№№ 22	*06*		
№№ 00	*21*	*LXIH , NN20*	Загрузить адрес *№№ 20* в пару *HL*– указателя адреса
№№ 01	*20*		Содержимое второго байта передается в младший регистр *L*регистровой пары /*HL*/, содержимое третьего байта передается в старший регистр /*H*/
№№ 02		*№№*	
№№ 03	*7E*	*MOVA , M*	Загрузить первое число в ячейке памяти *№№ 20* в аккумулятор
№№ 04	*23*	*INX H*	Инкрементировать пару *HL* до *№№ 21*
№№ 05	*86*	*ADD M*	Сложить второе число в ячейке памяти *№№ 21* с содержимым аккумулятора

№№ 06	*23*	*INX H*	Инкрементировать пару *HL* до *№№ 22*
№№ 07	*86*	*ADD M*	Сложить второе число в ячейке памяти *№№ 22* с содержимым аккумулятора
№№ 08	*23*	*INXH*	Инкрементировать пару *HL* до *№№ 23*
№№ 09	*77*	*MOVM, A*	Поместить содержащую в аккумуляторе сумму в ячейку памяти *№№ 23*
№№ 0A	*76*	*HLT*	Остановить МП

Если возникнет необходимость уменьшить на *1* содержимое регистровой пары, используется команда *DCXrp*, при необходимости уменьшить на *1* содержимое регистра -команда *DCRr*. Например: Уменьшить на *1* содержимое регистра *C13*

№№ 00	*0E*	*MVIC, 13*	Загрузить регистр *C* числом *13*
№№ 01	*13*		
№№ 02	*0D*	*DCRC*	Уменьшить на *1* содержимое регистра *C*
№№ 03	*76*	*HTL*	Останов

Микропроцессор может проводить арифметические операции с числами с двойной длиной слова. Сложение 16-разрядных чисел осуществляет частями, по восемь разрядов каждая, суммированием сначала младших разрядов чисел /крайних восьми разрядов справа каждого числа/, запоминая сигнал окончательного переноса из старшего для этой части разряда /значениями которого могут быть только *0* или *1*/ в одноразрядном регистре микропроцессора, называемом флагом переноса, и сложением затем этого числа со следующими восемью разрядами суммируемых чисел.

При сложении с переносом используется команда АДС. Применяется в тех случаях, когда базовой длины слова микропроцессора недостаточно и приходится вводить многоразрядные числа с последовательной обработкой отдельных байтов. В памяти такие числа хранятся в смежных ячейках и адресуются по младшему байту. Перед сложением самых младших байтов /при первом шаге сложения/ флаг переноса очищается с помощью команды *XRAr*, где "*r*"-название регистра.

Операция сложения чисел *17F5* и *341I* будет выглядеть так:

Старший байт	Флаг С	Младший байт	Числа
00010111_2		11110101_2	$17F5_{16}$
+		+	+
00110100_2		00010001_2	3411_{16}
+ 1	←1		
01001100_2		00000110_2	4006_{16}

Ее программа:

№№20		17	
№№21		F5	
№№22		34	
№№23		11	
№№00	21	LXIH NN21	Записать в регистровую пару HL адрес первого числа №№21
№№01		21	
№№02		№№	
№№03	01	LXIB NN23	Записать в регистровую пару BC адрес второго числа №№23
№№04		23	
№№05		№№	
№№06	AF	XRA A	Очистить аккумулятор и флаг переноса
№№07	OA	LDAX B	Загрузить аккумулятор числом, адрес которого указан в регистровой паре BC
№№08	86	ADDM	Сложить содержимое аккумулятора с числом, адреса которого указан вHL
№№09	02	STAX B	Содержимое аккумулятора разместить в ячейку, адрес которая в BC
№№07	2B	DCXH	Уменьшить на 1 содержимое пары HL
№№0A	3A	LDA, NN22	Загрузить аккумулятор числом, находящимся по адресу№№22
№№0B		22	
№№0C		№№	
№№0E	8E	ADCM	Сложить содержимое аккумулятора с числом, адрес которого указан в HL
№№0F	32	STANN22	Запомнить содержимое аккумулятора в ячейке с адресом№№22
№№10		22	
№№11		№№	
№№12	76	HLT	Останов

Микропроцессор не приспособлен для прямого вычитания. Чтобы его осуществить, он производит сложение, представляя вычитаемое вформе дополнительного кода и затем складывая его, пренебрегая переполнением. Дополнительный код получается прибавлением единицы к обратному /инверсному/ коду двоичного числа. А обратный код /инверсия/ двоичного числа образуется заменой в нем нулей единицами, а единиц нулями. Чтобы иметь возможность обрабатывать как положительные, так и х отрицательные числа, старший бит каждого байта интерпретируется как знаковый /0 -положительное, 1-отрицательное число/, а все остальные биты используются для предоставления числового значения.

Например, двоичное число 00000001_2, вычитается из 00001001_2 /09_{16} - $01_{16}=08_{16}$/. Преобразуем вычитаемое в дополнительный код.

00000001_2

инверсия 11111110_2 Добавить 1 11111110_2 +1

11111111_2 Дополнительный код

Дополнительный код вычитаемого 111111111 складывается с уменьшаемым, что дает сумму

$$\begin{array}{r} 00001001_2 \\ 11111111_2 \\ \hline 100001000_2 \end{array}$$

В старшем бите суммы 1 является переполнением и не принадлежит разности 00001000_2. Микропроцессор использует это переполнение для установления индикатора переноса. Вычитая, микропроцессор инвертирует переполнение, и результат становится содержимым индикатора "Флаг пере- носа". Переполнение 1 инвертируется и сбрасывает индикатор "флат переноса" в 0. Когда в ходе вычитания индикатор "флаг переноса" сбрасывается, это значит, что переноса не было и что первое число больше второго. Операция вычитания однобайтового числа не вызывает затруднений, так как микропроцессор имеет команды:

SUI - из содержимого аккумулятора вычитается второй байт команды. Результат помещается в аккумулятор.

SUBr - содержимое регистра с вычитается из содержимого аккумулятора. Результат помещается в аккумулятор SUBM - содержимое ячейки, адрес которой указан регистровой паре HL, вычитается из аккумулятора. Результат помещается аккумулятор.

"Флаг переноса" в командах SUI, SUB устанавливается, когда уменьшаемое меньше вычитаемого.

Пример1. Вычесть число 0С из числа ОЕ, пользуясь командой SUI.

№№ 00	3Е	MVIA,OE	Загрузить регистр аккумулятора числом ОЕ
№№ 01	ОЕ		
№№ 02	D6	SUI,OC	Из содержимого аккумугятора вычесть число ОС
№№ 03	ОС		
№№ 04	76	HLT	Останов

Пример 2. Иа числа 05 вычесть число Сб:

№№ 00	26	MVI H,06	Загрузить регистр числом О6
№№ 01	06		
№№ 02	3Е	MVIA,05	Загрузить аккумулятор числом 05
№№ 03	05		
№№ 04	94	SUB H	Из содержимого аккумулятора вычесть содержимое регистра Н
№№ 05	76	HLT	Останов

Процедура вычитания многобайтовых чисел напоминает сложение с использованием ADC, но требует применять команду вычитания с заемом S

Вычитание проводится по байтам начиная с младших. При вычитании младших байтов чисел необходимо применять команду SUB, а для вычитания остальных-команду SBB, которая будет учитывать состояние разряда "флага переноса".

Пример 3. Решить задачу 253F -194 F:

№№ 00	3E	MVIA,3F	Загрузить регистр аккумулятора числом 3F
№№ 01	3F		
№№ 02	06	MVI B,4F	Загрузить регистр В числом 4
№№ 03	4F		
№№ 04	90	SUBB	Вычесть из содержимого аккумулятора содержимое регистра В
№№ 05	32	STA NN00	Запомнить и занести в ячейкупамяти с адресом №№ 20
№№ 06	20		
№№ 07	NN		
№№ 08	I6	MVID, 2E	Загрузить регистр Д числом 2E
№№ 09	2E		
№№ 0A	7A	MOV AD	Передатьсодержимое регистраДв аккумулятор
№№ 0B	IE	MVI E,19	Загрузить регистр Е числом 19
№№ 0C	19		
№№ 0Д	9B	SBBE	Вычесть из содержимого аккумулятора содержимое регистра В
№№ 0E	32	STA №№21	Запомнить и занести в ячейку памяти с адресом NN21 содержимое аккумулятора
№№ 0F	21		
№№ 10	№№		
№№ 11	76	HLT	Останов

ЗАДАНИЕ К ПРАКТИЧЕСКОЙ РАБОТЕ

1. Разработать программы для следующих задач:

а/ сложить четыре произвольных однобайтовых числа, расположенных в последовательных ячейках, и из сумм вычесть сумку двух других произвольных однобайтовых чисел;

б/ сложить два произвольных многобайтовых числа, расположенных не в соседних ячейках, и из их суммы вычесть произвольное однобайтовое число;

в/ повторить пункт "б", но при этом выбрать уменьшаемое меньше вычитаемого;

г/ полученные результаты проанализировать и объяснить.

СОДЕРЖИМОЕ ОТЧЕТА

Отчет должен содержать: название работы, цель, перечень команд арифметических действия с комментариями, программ, разработанные и исследованные согласнао заданию, решения задач, выполненные в двоичных числах, а такое анализ подученных результатов и выводы.

ЗАДАНИЕ ДЛЯ САМОПРОВЕРКИ

1. С помощью каких команд осуществляется сложение чисел?
2. Какая из команд сложения является однобайтовой?
3. Что означает команда INX?
4. Когда используется команда XRA?
5. Что такое сложение с переносом и какая команда используется в этом случае?
6. Как осуществляется сложение 16-разрядных чисел?
7. Что означает команда ADDM?
8. В каких случаях используется команда ADI?
9. Как в микропроцессоре осуществляется вычитание?
10. Что такое дополнительный код?
11. Как понимать слово "инверсия"?
12. В каком случае используется команда SBB?
13. Какую команду используют при вычитании из содержимого аккумулятора содержимое регистра?
14. Чем отличаются команда SUB от SBB ?
15. Для чего нужен индикатор "флаг переноса"?
16. В каких случаях на индикаторе "флаг переноса" появляется 1?
17. Что означает переполнение регистра?
18. Как производился вычитание двоичных чисел?

Практичні заняття № 5
ЛОГИЧНІ ОПЕРАЦІЇ, МАСКУВАННЯ ДАНИХ

Цель работы - изучить логические операции, программные способы маскирования данных и тестирования заданного состава бит.

Во многих случаях при выполнении программ необходимо проверять или изменять / маскировать/ состояние одного или нескольких разрядов числа в аккумуляторе. Это можно осуществить с помощью следующих команд логических операций : И, Или, ИЛИ ИСКЛАНАЩЕЕ, НЕ /инверсия/ и сдвига. Здесь именно аккумулятор составляет ядро большинства операций. Как и при арифметических командах, способ адресации способ и место нахождения других данных здесь влияет на системе.

Логические операции являются поразрядными, т.е. выполняются независимо для всех 8 бит операндов. Не адресуемый операнд находится в

аккумуляторе, туда же загружается результат операции. По результату операции модифицируется состояние всех флажков, кроме флажка переноса С, который никогда не может быть установлен в единицу и поэтому принудительно сбрасывается.

Операция И -это операция логического умножения, поэтому она часто обозначается так же, как произведение в элементарной алгебре. Эту операцию называют конъюнкцией и обозначат знаком «^» .Поэтому выражения y=A^B; y=A*B; y=A÷B;y=AB эквиваленты и читаются одинаково: "игрек равен А и В".

В схеме И сигнал на выходе элемента, соответствующий логической 1, появляется только в том случае, когда аналогичные сигналы присутствуют одновременно на входах А и В. Таблица истинности для операции И имеет вид:

A	B	A и B
0	0	0
0	1	0
1	0	0
1	1	1

Операция И выполняется е помощью регистров адресации посредством мнемокоманды ANAr. При выполнении ее производится поразрядная конъюнкция операндов в аккумуляторе и регистре общего назначения. В качестве примера рассмотрим операцию логического умножения чисел 13^01. Первое из них поместим в аккумулятор, загрузив его вторым байтом команды MVIA, а второе поместим вторым байтом команды MVID в регистр Д.

№00	16	MVID,01	Загрузить регистр Д числом 01
№01	01		
№02	3E	MVIA,13	Загрузить аккумулятор числом 13
№04	13		
№05	A2	ANA D	Выполнить логическое умножение
№06	76	HLT	Останов

Здесь содержимое аккумулятора /13_{16}– 00010011_2/ подвержено операции И побитно с чиклом /01_{16} =000000001_2/ в регистре Д. Согласно таблице истинности результатом логического перемножения этих чисел будет 00000001_2. Оно помещается в аккумулятор. Результатом всех операций И будет сброс индикатора "флаг переноса". Результат операции И проверяется с целью определения не нуль ли он, если нет то индикатор "флаг нуля" сбрасывается 0.

Из примера видно, что второй опоранд-00000001 используется для сброса в 0 семи старших бит, т.е. с помощью команды ANA, используя слово-маску, можно сбрасывать в 0 определённые биты слова в аккумуляторе, оставляя другие неизменимыми. Маска может быть использована с учетом наличия индикатора "флаг нуля" для тестирования значений 0 или I в любом из разрядов числа. Так, в приведенном примере для тестирования значения 0 или 1 в позиции младшего бита аккумулятора на единственную 1 надевается маска 00000001, а если

необходимо протестировать на единицу 2-, 3- и 5- битовые числа в аккумуляторе, используется маска 00010110.

Таким образом, логическое умножение числа в аккумулятора и маски очищает разряд числа, если в соответствующем разряде маски записан 0, и не изменяет его, если в разряде записана 1.

Команда И может осуществляться с помощью непосредственной адресации. При этом ее мнемоника ANI означает: выполнить логическое умножение содержимого аккумулятора и второго байт команды. При косвенной регистровой адресации используется команда ANA M. где M - содержимое регистровой пары HL /адреса ячейки/.

Операция ИЛИ - это операция логического сложения и поэтому обозначается знаком "+". Её называет также дизъюнкцией и чаще обозначают знаком «v». Таким образом, выражения X=A+B и X=AvB эквивалентны и читаются как "икс" равен A или B.

В схеме ИЛИ сигнал на выходе элемента, соответствующий логической 1, появляется в случае присутствия единичного сигнала хота бы на одном из входов A или B.

Таблица истинности для операции ИИ имеет следующий вид:

A	B	A	ИЛИ	B
0	0		0	
0	1		1	
1	0		1	
1	1		1	

Команда ИЛИ по аналогии с командой И может выполняться:

с непосредственной адресацией:

ORI - выполнить логическое сложение содержимого аккумулятора и второго байта команды;

с регистровой адресацией:

ORAr - выполнить операцию логического сложения содержимого аккумулятора и регистра;

с косвенной регистровой адресацией:

ORAM- выполнить операцию логического сложения содержимого аккумулятора с содержимым ячейки, адрес которой указан в регистровой паре HL.

Команды ORA, ORI осуществляют поразрядную дизъюнкцию операндов. Эти команды применяются для расстановки определенных битов слова в аккумуляторе с помощью слова-маски, а также установки слова из полей других слов. Например, результатом операции ORA с операндами 00001111 и 11110000 будет установленное слово 11111111. В качестве примера рассмотрим логическое сложение, чисел $CC_{16}v0F_{16}$, из которых второе является числом-маской;

$CC_{16} = 11001100_2$ Результат:

$0F_{16} = 00001111_2$ $11001111_2 = CF_{16}$

Программа данного примера имеет вид:

№00	3E	MVIA,CC	Загрузить аккумулятор числом CC
№01	CC		
№02	3E	MVIB,0F	Загрузить аккумулятор числом OF
№03	0		
№04	BO	ORA B	Выполнить операцию логического сложения аккумулятора и регистра B
№05	76	HLT	Останов

В данном примере числа подвержены операции ИЛИ также побитно согласно таблице истинности ИЛИ. Содержимое 00001111 регистра B может рассматриваться как маска, которая всегда будет обращать 4 младших бита в 1111.

Команда ИСКЛЮЩЕЕ ИЛИ с регистровой адресацией имеет мнемонику XRAr, производит поразрядное двоичное сложение операндов. Эта операция обозначается знаком (+).Так, запись A(+) B означает либо A, либо B, но не оба вместе. Таблица истинности для операции ИСКЛЮЧАЮЩЕЕ ИЛИ имеет вид:

A	B	A	ИЛИ	B
0	0		0	
0	1		1	
1	0		1	
1	1		0	

Команда XRA применяется для инвертирования определенных битов слова с помощью слова-маски на основе тождества 1(+)X=X. Произведем операцию ИСКАЮЩЕГО ИЛИ чисел FF_{16} (+) 50_{16}.

$FF_{16}=11111111_2$ Результат:

$50_{16}=01010000_2$ $101011111_2=AF_{16}$

Программа имеет вид:

№№20	FF		Записать число FF в ячейку №№ 20
№№ 21	50		Записать число 50 в ячейку №№ 21
№№00	3A	LDA NN20	Загрузить аккумулятор содержимым
№№01		20	ячейки, адрес которой н 20
№№02		№	
№№03	21	LXIH, NN21	Загрузить регистровую пару HL 21 числом

№№04		21	
№№05		№	
№№06	46	MOV B, M	Передать информацию из памяти в B
№№07	A8	XRAB	Выполнить операцию ИСКЛЮЧАЮЩЕЕ ИЛИ B
№08	32	STA NN22	Записать содержимое аккумулятора в память по адресу № 22
№№09		22	
№№0A		№№	
№№0B	75	HLT	Останов

Другое применение команды XRA связано со сравнением слов на абсолютное равенство. В единственном случае, когда операнды поразрядно совпадают, результат операции содержит нули во всех разрядах /согласно тождеству "X + X=0"/.

Выполнение операции ИСКАЮЩЕЕ ИЛИ любого числа с самим собой всегда дает результат 00000000, при этом индикатор "флаг нуля" устанавливается в 1, что означает нулевое содержание индикаторов, а индикатор "флаг переноса" всегда будет сброшен 0. Таким образом, для того чтобы очистить регистр, необходимо выполнить операцию XRA r с r.

Например, очистить аккумулятор от загруженного в него числа EC:

№00	3E	MVIA,13	Загрузить аккумулятор числом EC
№01		EC	
№03	AF	XRAA	Очистить аккумулятор
№04	76	HLT	Останов

Двухбайтовая команда с непосредственной адресацией логической операции ИСКЛЮЧАЮЩЕЕ ИЛИ имеет мнемоническое обозначение XRI.

При косвенной регистровой адресации используется команда XRAM

Операция НЕ, или операция инверсии, обозначается черточкой над инвертируемой величиной. Например, НЕ А записывается как «$\overline{A}$». Если А есть 0, НЕ А должно быть 1, и наоборот.

Таблица истинности для операции НЕ будет иметь следующий вид:

A	A
0	1
1	0

Операция инвертирования осуществляется только с содержимым аккумулятора посредством мнемокоманды CMA - инвертировать А. Рассмотрим программу, использующую CMA:

№№20	CE		Записать число CE в ячейку №№ 20
№№00	21	LXIH, NN20	Записать регистровую пару HL в адрес №№ 20
№№01		20	
№№02		№№	
№№03	7E	MOV A,M	Получить число из адреса, указанного в HL

№№04	2F	CMA	Инвертировать число в аккумуляторе
№№05	23	INX M	Увеличить на 1 адрес в регистрах HL
№№06	77	MVI M, A	Записать число из аккумулятора по адресу в HL
№№07	E7	RST4	Прервать выполнение программы

Команды сравнения

Команда сравнения CM производит вычитание значения адресуемого операнда из содержимого аккумулятора и модифицирует по результату значения всех разрядов состояния, но при этом не изменяет, содержимого аккумулятора, а следовательно, не заносится в него. Результат фиксируется регистром признаков с помощью индикатора Z "флаг нуля". Если числа одинаковые, Z=1, если нет-Z= 0.

В отличие от команды CMP, команда CMPM использует косвенную регистровую адресацию, т.е. производит вычитание содержимого регистровой пары HL из содержимого аккумулятора. Команда CPI использует непосредственную адресацию. По этой команде вычитается операнд, записанный вторым байтом, из содержимого аккумулятора. В качестве примера рассмотрим программу:

№№20	F4		Записать число F4 в ячейку № 20
№№21	06		Записать число 06 в ячейку № 21
№№00	21	LXIH, NN20	Загрузить регистровую пару HL числом№ 20
№№01		20	
№№02		№№	
№№03	06	MVIB,06	Загрузить регистр B числом 06
№№04	06		
№№05	7E	MOVA,M	Загрузить аккумулятор числом, адрес которого в HL
№06	B8	CMPB	Сравнить содержимое в регистре B с A
№07	23	INX M	Увеличить на единицу содержимое пары HL

№№ 08	7E	*MOVA, M*	Загрузить аккумулятор числом, адрес которого в HL
№№ 09	B8	*CMPB*	Сравнить содержимое BcA
№№ 0A	76	*HLT*	Останов

Результаты сравнения оцениваются по состоянию индикатора «флаг нуля».

Команды сдвига

Команды сдвига требуют 1 байт памяти. Операндом их является содержимое аккумулятора, в котором формируется результат. Сдвиги выполняются влево и вправо только на один разряд. В зависимости от того, что помещается в освобождающийся при сдвиге бит и как используются

выдвигающийся (спадающий) бит и флажок переноса, вводится несколько команд сдвигов.

В командах циклического сдвига (ротациях) влево *RLC*и вправо *RRC*выдвигающийся бит помещается в освобождающийся и, кроме того, фиксируется на флажке переноса С.

Например:

№№ 10	94		Записать число 94 в ячейку №№ 10
№№ 00	21	*LXIH, NN10*	Загрузить регистровую пару HL адресом №№ 10
№№ 01		10	
№№ 02		№№	
№№ 03	7E	*MOVA, M*	Загрузить в аккумулятор число, адрес которого в HL
№№ 04	07	*RLC*	Сдвинуть влево содержимое А
№№ 05	76	*HLT*	Останов

В командах сдвига через перенос влево *RAL*и вправо *RAR*выдвигающийся бит помещается во флажок переноса С, а текущее значение флажка С передается в освобождающийся бит.

Например:

№№ 00	AF	*XRAA*	Очистить флаг переноса
№№ 01	3E	*MVIA, 31*	Загрузить аккумулятор числом 31
№№ 02	31		
№№ 03	1F	*RAR*	Сдвинуто вправо содержимое аккумулятора через разряд переноса
№№ 04	76	*HLT*	Останов

В этом примере содержимое аккумулятора (31_{16} = 00110001_2) сдвинуто на одну позицию вправо и его младший бит (здесь 1) передается в позицию бита индикатора переноса, тогда как имевшийся там бит занимает позицию старшего бита аккумулятора, в котором содержится 0001100 после завершения операции. Индикатор переноса установится в 1, индикатор нуля не изменится.

Используя одну или несколько команд циклического сдвига, можно тестировать весь заданный состав бит.

ЗАДАНИЕ К ПРАКТИЧЕСКОЙ РАБОТЕ

На ассемблере решить следующие задачи:

а) сбросить 4 старших разряда регистра А при неизменном состоянии остальных разрядов;

б) сформировать дополнительный код содержимого регистра А и передать его в регистр В;

в) сформировать логическое выражение /НЕВ/ИС/ИЛИД/ в регистре А;

г) загрузить в 4 старших разряда регистра A код 0101, не изменяя содержимого остальных разрядов;

д) передать 4 младших разряда регистра B в 4 младших разряда регистра D и 4 старших разряда регистра E. Старшие биты регистров D и E должны быть сброшены;

ж) число в регистре D сравнить с тремя числами, записанными в последовательные ячейки памяти;

з) составить программу по приведенной функциональной структурной схеме:

СОДЕРЖАНИЕ ОТЧЕТА

Отчет должен содержать название и цель работы, полный перечень логических команд с комментариями, результаты исследований приведенных в инструкции программ, а также программы, разработанные и исследованные в процессе выполнения задания к практической работе, и выводы.

ЗАДАНИЕ ДЛЯ САМОПРОВЕРКИ

1. Что происходит в результате логического умножения числа в аккумуляторе и маски?
2. Что означает понятие «маскирование данных»?
3. Как осуществляется операция логического сложения числа в аккумуляторе и маски?
4. Что происходит в результате выполнения команды $ORA\ r$?
5. Чем отличаются команды ANA, ORA, XRA от команд ANI, ORI, XRI ?

6. Какую команду следует использовать при необходимости инвертировать число?
7. Чем отличаются команды RAL и RLC ?
8. Что общего между командами RAR и RRC ?
9. Что означает запись / A /(+)/ r / ?
10. Что такое конъюнкция и дизъюнкция?
11. В чем отличие операций ИЛИ и ИСКЛЮЧАЮЩЕЕ ИЛИ?
12. Что осуществляется по команде CMP?
13. Где фиксируется результат сравнения двух чисел?
14. Чем отличается команда CMP от команды CMI?

Практичні заняття № 6

ОРГАНІЗАЦІЯ ПЕРЕХОДІВ В ПРОГРАМАХ

/РОЗГАЛУЖЕННЯ ПРОГРАМ І ЦИКЛИ/

Цель работы – изучить команды ветвления и перехода и разработку программ с условной и безусловной передачей управления.

Обычно микропроцессор выполняет команды последовательно. 16-разрядный счетчик команд типового микропроцессора хранит всегда адрес следующей извлекаемой из памяти команды до её выполнения. Содержимое его обычно повышается в каждом счете. Однако линейные прикладные программы на практике почти не встречаются. В разветвляющихся и циклических программах и при использовании подпрограмм приходится выполнять не следующую по порядку команду, а команду, находящуюся в другой ячейке программной памяти. Для этого достаточно загрузить в программный счетчик адрес новой ячейки, называемый адресом перехода. Такая процедура называется передачей управления, а специальные команды, которыми она реализуется, - командами передачи управления или командами управления программой.

Эти команды разделены на две группы: безусловного перехода и условного перехода.

Команда безусловной передачи управления без возврата*IMP*, называемая также безусловным переходом или просто переходом, состоит из байта кода операции и двух байтов полного 16-битного адреса перехода. Младшая часть адреса находится во втором байте памяти, старшая – в третьем. При выполнении команды *IMP*адрес перехода загружается в программный счетчик, а его текущее содержание теряется. Поэтому при осуществлении безусловного перехода одна команда *IMP*будет использована для запуска счетчика команд в момент начала выполнения новой программы, а вторая – для возвращения к выполнению прерванной программы.

Применение безусловных переходов рассмотрим на примере следующей программы:

№№ 20	05		Записать число 5 в ячейку №№ 20
№№ 21	06		Записать число 6 в ячейку №№ 21
№№ 22	03		Записать число 3 в ячейку №№ 22
№№ 00	21	*LXIH, NN20*	Записать в регистровую пару *HL* число №№ 20
№№ 01		20	
№№ 02		№№	
№№ 03	01	*LXIB, NN22*	Загрузить регистровую пару *BC* числом №№ 22
№№ 04		22	
№№ 05		№№	
№№ 06	OA	*LDAxB*	Загрузить аккумулятор содержимым ячейки, адрес которой указан в *BC*
№№ 07	86	*ADDB*	К содержимому аккумулятора прибавить число, адрес которого указан в *HL*
№№ 08	C3	*IMPNN10*	Безусловный переход по адресу №№ 10
№№ 09		10	
№№ OA		№№	
№№ OB	7E	*MOVA, M*	В аккумулятор загрузить содержимое ячейки, адрес которой указан в *HL*
№№OC	00	*NOP*	Пустая операция
№№OD	C6	*ADI* 02	К содержимому аккумулятора прибавить число 02
№№OE		02	
№№ OF	76	*HLT*	Останов
№№ 10	32	*STANN23*	Записать содержимое аккумулятора в память по адресу №№ 23
№№ 11		23	
№№ 12		№№	
№№ 12	C3	*IMPNNOB*	Безусловный переход по адресу №№ OB
№№ 14		OB	
№№ 15		№№	

Организация условных переходов в микроЭВМ осуществляется с помощью регистра признаков микропроцессора.

Регистр признаков /флагов/ имеет пять разрядов, каждый из которых устанавливается по определенному правилу в соответствии с выполнением

микропроцессором последней программы. Этими разрядами являются следующие.

1. Разряд /флаг/ переполнения C. В него записывается 1, если при выполнении арифметической команды или команды сдвига было переполнение аккумулятора, в противном случае в разряд записывается 0.
2. Разряд /флаг/ знака S. В него записывается 1, если при выполнении арифметической или логической команды в старшем восьмом разряде аккумулятора записана 1, в противном случае в разряд записывается 0.
3. Разряд /флаг/ нулевого результата Z. В него записывается 1, если при выполнении арифметической или логической команды во всех разрядах числа в аккумулятора имеется 0, в противном случае в разряд записывается 0.
4. Дополнительный разряд /флаг/ переполнения AO. В него записывается 1, если при выполнении команд в аккумуляторе возникает единица переноса из третьего разряда числа.
5. Разряд /флаг/ четности P. В него записывается 1, если при выполнении команды количество единиц в разрядах аккумулятора будет четным.

Трехбайтовые команды условной передачи управления без возврата, называемые также условными переходами и разветвлениями, осуществляет передачу управления только при удовлетворении некоторого условия, заданного в коде операции. Если условие не удовлетворяется, то передача управления не происходит, а выполняется следующая по порядку команда, т.е. в этой ситуации команда разветвления эквивалентна холостой команде.

Проверяемым условием является текущее значение одного из флажков указываемое в коде операции четвертым и пятым битом. При этом могут проверяться результаты операции: нулевой или ненулевой, положительный или отрицательный, четный или нечетный результаты, отсутствие или наличие переноса.

Команды условных переходов:

INC – передает управление по нулевому значению разряда переноса, C=0;

IC – передает управление по единичному разряду переноса, C=1;

IZ – передает управление по нулевому значению результата, Z=1;

INZ – передает управление по нулевому значениюрезультата, Z=0;

IP – передает управление по положительному значению результата;

IM – передает управление по отрицательному значению результата;

IPE – передает управление по четности кода результата;

IPO – передает управление по нечетности кода результата.

Действие команд INC иIC рассмотрим при пошаговом выполнении следующей программы:

№№ 20	0F		Записать число 0Fв ячейку №№ 20
№№ 00	06	*MVIB,* 02	Загрузить регистр Вчислом 02
№№ 01	02		
№№ 02	3E	*MVIA, FF*	Загрузить аккумулятор числом FF
№№ 03	FF		
№№ 04	80	*ADDB*	К содержимому А прибавить В
№№ 05	DA	*ICNN* 12	Условный переход по адресу №№ 12,
№№ 06	12		если в результате появится единица
№№ 07	№№		переноса, т.е. С=1
№№ 08	78	*MOVA, B*	Содержимое В передать в А
№№ 09	21	*LXIHNN* 20	Загрузить регистровую пару*HL* адресом ячейки памяти №№ 20
№№ 0A	20		
№№ 0B	№№		
№№ 0C	46	*MOVB, M*	К содержимому Вприбавить число в ячейке №№ 20
№№ 0D	90	*SUBB*	Из содержимого А вычесть содержимое В
№№ 0E	D2	*INCNN00*	Передает управление по адресу №№ 00, если после операции *SUBB*разряд переноса С=0
№№ 0F	00		
№№ 10	№№		
№№ 11	76	*HLT*	Останов
№№ 12	C3	*IMPNN 0B*	Безусловный переход по адреса №№ 08
№№ 13	08		
№№ 14	№№		

В данной программе после команды *ADDB*суммирования FF_{16} + 02_{16} использована команда перехода *IC*по адресу №№ 12 при условии, если в результате сложения появится единица переноса, т.е. С = 1. Поскольку перенос имеется:

$FF_{16} = 11111111_2$	флаг С	Результат
+		
$02_{16} = 00000010_2$	1	00000001_2

управление передается по адресу №№ 12, где записана команда безусловного перехода IMPпо адресу №№ 08 /*MOVA, B*/. В результате этой операции число 02_{16} пересылается из регистра Вв аккумулятор. Далее посредством команд *LXIH*и *MOVB, M*в освободившийся регистр производится вычитание $02_{16} – 0F_{16}$ по команде *SUBB*. После проверки условия нулевого значения разряда переноса, С = 1, управление может быть передано либо по адресу №№ 20, либо

по адресу №№ 11. Если С = 0, в счетчик команд будет записан адрес №№ 00, если С = 1 – адрес №№ 11.

Поскольку в результате операции *SUB* $02_{16} - 0F_{16}$ образуется заем/т.е. С = 1/

$02_{16} = 00000010_2$ флаг С 00000010_2

\-

$0F_{16} = 00001111_2$ 1 11110001_2 – дополнительный код

11110011_2 – результат

то управление передается только по адресу №№ 11. Здесь следует обратить внимание на то, что результат вычитания отрицательный и по этой причине разность чисел представляется в форме дополнительного кода.

Рассмотрим действие условных переходов по признаку нулевого значения результата IZ, т.е. когда Z = 1 на примере выполнения в пошаговом режиме программы:

№№ 20	01		Записать число 01 в ячейку №№ 20
№№ 21	02		Записать число 02 в ячейку №№ 21
№№ 22	03	*MVIA, 04*	Записать число 03 в ячейку №№ 22
№№ 23	04		Записать число 04 в ячейку №№ 23
№№ 00	3E		Загрузить аккумулятор числом 04
№№ 01	04		
№№ 02	21	*LXIHNN20*	Загрузить регистровую пару *HL* адресом №№ 20
№№ 03	20		
№№ 04	№№		
№№ 05	01	*LXIB, 0001*	Загрузить регистровую пару *BC* числом 0001
№№ 06	01		
№№ 07	00		
№№ 08	56	*MOVD, M*	Загрузить регистр *D*числом, адрес которого в *HL*
№№ 09	92	*SUBD*	Вычесть Dиз A

№№ 0A	CA	*IZ, NN11*	Условный переход по адресу №№ 11, если в результате операции *SUBD* результат равен нулю, т.е. Z=1
№№ 0B	11		
№№ 0C	№№		
№№ 0D	09	*DADB*	Сложить содержимое регистровых пар *HL*и*BC*
№№ 0E	C3	*IMP, NN08*	Безусловный переход по адресу №№ 08
№№ 0F	08		
№№ 10	№№		
№№ 11	76	*HLT*	Останов

Программа построена на использовании циклических процессов с проверкой результата на каждом цикле. Цикл построен следующим образом. Из ячейки памяти с адресом №№ 20 переписывается ее содержимое в регистр Dкомандой пересылки *MOVD, M*. Это команда с косвенной регистровой адресацией, поэтому здесь используется регистровая пара *HL*. Затем операцией *SUBD*из содержимого регистра аккумулятора вычитается содержимое регистра D, после чего сразу же проверяется результат посредством команды условного перехода *IZ*. Если результат равен нулю /*Z* = 1/, то управление передается по адресу №№ 11 и здесь работа микроЭВМ заканчивается. Если же обнаруживается неравенство нулю результата операции вычитания /*Z* = 0/, то управление передается на повторение описанного цикла – по адресу 0D, по которому в программе значится операция *DADB*–сложить содержимое регистровой пары*HL*с содержимым регистровой пары *BC*. Так как в регистровую пару *BC*загружено число 0001, то на каждом цикле адрес ячейки памяти, содержащийся в регистровой паре *HL*, модифицируется. На первом цикле происходит №№ 20 + 0001 = №№ 21, на втором №№ 21 + 0001 = №№ 22 и так далее.

В целом вычислительный циклический процесс выглядит следующим образом:

$$04_{16} - 01_{16} = 03_{16}$$

$$04_{16} - 02_{16} = 02_{16}$$

$$04_{16} - 03_{16} = 01_{16}$$

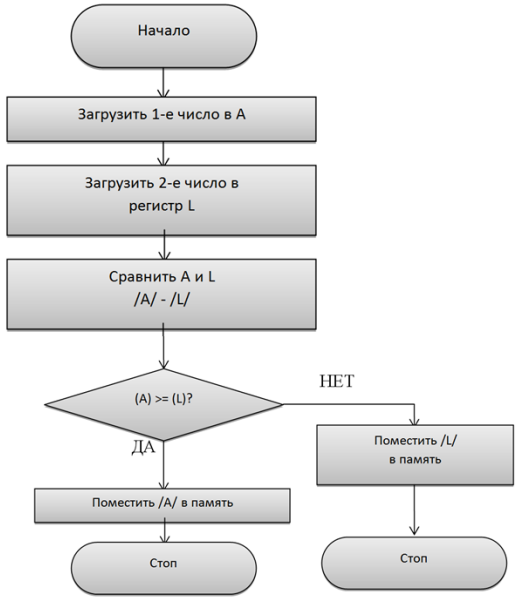
$$04_{16} - 04_{16} = 0$$

Для обеспечения цикличности (т.е. непрерывности вычислений) после команды *DADB*в программе использована команда безусловного перехода *IMP*по адресу №№ 08.

Обычно перед составлением программы разрабатывается решение задачи. Чаще оно выполняется в виде функциональной структурной схемы, где каждая рамка схемы соответствует одной эквивалентной команде.

Рассмотрим простую задачу сравнения двух чисел. Нужно найти наибольшее и поместить его в определенную ячейку памяти.

Структурная схема решения этой задачи имеет вид:



В двух первых прямоугольниках представлена операция загрузки двух чисел в регистры АиL. Следующий прямоугольник соответствует операции сравнения, где содержимое регистра Lвычитается из содержимого регистра А. Команды сравнения не разрушают содержимого регистров, но влияют на индикатор. Следующий ромб называется знаком принятия решения. Он завершает в программе этап решения и содержит поставленный вопрос [/A/ >= /L/?]. Если ответ на этот вопрос «Да», программа продолжается последовательно и содержимое регистра А помещается в память, затем процессор останавливается. Если «Нет», программа ответвляется в сторону и содержимое регистра Lпомещается в память, затем МП останавливается.

Программа реализующая это решение для чисел 0Fи 06, имеет вид:

№№ 00	3E	*MVIA, 0F*	Загрузить в аккумулятор число 0F
№№ 01		0F	
№№ 02	2E	*MVIL, 06*	Загрузить в регистр Lчисло 06
№№ 03		06	

№№ 04	BD	CMPL	Сравнить /A/ и /L/. Индикатор C=1, если /A/ </L/
№№ 05	DA	IC, NN0C	Перейти на адрес №№ 0C, если C=1 (если A <L); если нет, продолжить последовательно
№№ 06		0C	
№№ 07		№№	
№№ 08	32	STA, NN00	Поместить /A/ в ячейку памяти №№ 00
№№ 09		00	
№№ 0A		№№	
№№ 0B	76	HLT	Останов
№№ 0C	7D	MOVA, L	Передать в аккумулятор
№№ 0D	32	STA, NN00	Поместить /A/ в ячейку памяти №№ 00
№№ 0E		00	
№№ 0F		№№	
№№ 10	76	HLT	Останов

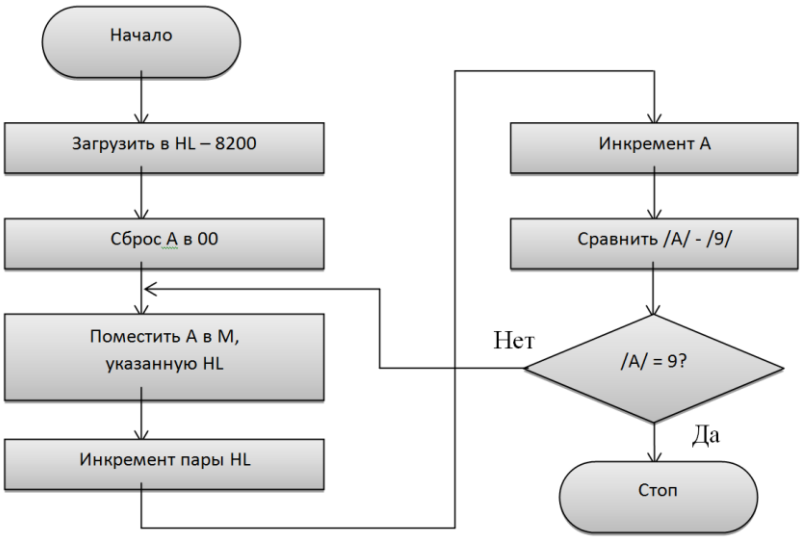
ЗАДАНИЕ К ПРАКТИЧЕСКОЙ РАБОТЕ

1. Приведенные в инструкции программы проанализировать.
2. Разобрать программу, по которой произвольный ряд чисел 01, 02, 03, 05, 07, 09, 0E, хранимый в ячейках памяти с последовательными адресами, будет складываться друг с другом до появления признака переполнения.
3. Разработать и исследовать циклическую программу размещения ряда чисел (от 0 до 8) последовательно в память с адресами от №№ 00 до №№ 08.

СОДЕРЖАНИЕ ОТЧЕТА

Отчет должен содержать название и цель работы, полный перечень логических команд с комментариями, результаты исследований приведенных в инструкции программ, а также программы, разработанные и исследованные в процессе выполнения задания к лабораторной работе, и выводы.

Структурная схема последовательного счета имеет вид:



Практичні заняття № 7

ОПЕРАЦІЇ ЗІ СТЕКОМ ТА ЗВЕРНЕННЯ ДО ПІДПРОГРАМ

Цель работы – освоение методов программирования с использованием стековой памяти и обращение к подпрограммам.

Типовой микропроцессор содержит указатель стека - специализированный 16-разрядный регистр-счетчик, содержимым которого всегда является адрес. Этот адрес принадлежит особой группе ячеек памяти данных, которая называется стеком.

Стек типового микропроцессора содержится в ОЗУ, и его положение определяется программистом с помощью команд *LXISP*– загрузить указатель стека двухбайтовым адресом или *SPHL*– загрузить указатель стека содержимым пары регистров *HL*. Только после этого подается команда *PUSH*– записать два байта данных в стек. Однако в микроЭВМ часто функция организации стековой области ОЗУ закреплена за программой *МОНИТОР*. Поэтому, если пользователь выберет в качестве начального адреса стековый адрес памяти, предусмотренный *МОНИТОРом*, то он располагает возможностью не предварять команды *PUSH*и *POP*командами*LXISP*и*SPHL*, т.е. может непосредственно использовать команды*PUSH*и*POP*.

Команда *PUSH*помещает содержимое регистровой пары в двух ячейках памяти стека. Последовательность событий при выполнении операции *PUSH*:

указатель стека декрементируется и показывает на новый адрес ячейки памяти. Старший байт из регистра данных *H*помещается в стек;

указатель стека снова декрементируется и показывает следующий адрес (на один меньше) ячейки памяти. Младший байт из регистра данных *L*загружается в стек. В стек может загружаться также содержимое аккумулятора и регистра состояния. При этом команда имеет вид *PUSHPSW*согласно которой:

до операции *PUSHPSW*указатель стека указывает на последнюю ячейку стека. Ее называют вершиной стека. Затем указатель стека декрементируется и указывает на ячейку памяти меньше на единицу. Содержимое аккумулятора загружается в стек по этому адресу;

указатель стека снова декрементируется и указывает на новую ячейку, номер которой меньше предыдущей на один. Содержимое регистра состояния загружается по этому адресу.

Стек продолжает расти, пока длится процесс загрузки в него.

Обычно каждой команде загрузки в стек (*PUSH*)позже будет соответствовать команда извлечения из стека (*POP*), по которой данные берутся из вершины стека. Поскольку стек является памятью типа «последний вошел – первый вышел», данные должны извлекаться из стека в порядке, обратном загрузке. Извлечение данных из стека и их восстановление в регистре адреса данных является действием, обратным операции загрузки в стек. Между командами *PUSH*и *POP*обычно располагаются команды, которые меняют данные, содержащиеся в регистрах микропроцессоров. Однако, после извлечения данных из стека первоначально записанная информация в регистрах восстанавливается.

В качестве примера проанализируем следующую программу:

№№ 00	3E	*MVIA*, 88
№№ 01	88	
№№ 02	21	*LXIH*, 4444
№№ 03	44	
№№ 04	44	
№№ 05	11	*LXID*, 5555
№№ 06	55	
№№ 07	55	
№№ 08	01	*LXIB*, 7777
№№ 09	77	
№№ 0A	77	*PUSHH*
№№ 0B	E5	*PUSHD*
№№ 0C	D5	
№№ 0D	01	*PUSHB*
№№ 0E	F5	*PUSHPSW*
№№ 0F	17	*RAL*
№№ 10	29	*DADH*

№№ 11 03 *INXB*
№№ 12 1B *DCX D*
№№ 13 F1 *POP PSW*
№№ 14 C1 *POP B*
№№ 15 D1 *POP D*
№№ 16 E1 *POP H*
№№ 17 76 *HLT*

Здесь аккумулятор, регистровые пары *HL*,*DE*,*BC*загружаются числами соответственно 88, 4444, 5555, 7777, а затем заносятся в стек. После этого содержимое аккумулятора и регистровых пар изменяется с помощью команд *RAL, DADH, INXB, DCXD*. При последующем извлечении из стековой памяти посредством команды *POP*содержимое аккумулятора и регистровых пар восстанавливается.

Выполним в пошаговом режиме работы микроЭВМ программу суммирования шестнадцатеричных чисел:

24 + 18 – 21 + AA + 11 – 88 – 10 + 51.

Первое из них загружается в аккумулятор, остальные – в стековую память. Первым в стек записано число 51, в последнем – 18. Это сделано для того, чтобы при извлечении из стека и выполнении арифметических операций не менять порядок следования чисел (слева направо).

№№ 00 01 *LXIB*, 0051
№№ 01 51
№№ 02 00
№№ 03 C5 *PUSHB*
№№ 04 01 *LXIB*, 0010
№№ 05 10
№№ 06 00
№№ 07 C5 *PUSHB*
№№ 08 01 *LXIB*, 0088
№№ 09 88
№№ 0A 00
№№ 0B C5 *PUSH B*
№№ 0C 01 *LXIB*, 0011
№№ 0D 11
№№ 0E 00
№№ 0F C5 *PUSH B*
№№ 10 01 *LXI B*, 00AA
№№ 11 AA
№№ 12 00
№№ 13 C5 *PUSH B*
№№ 14 01 *LXI B*, 0021

№№ 15	21	
№№ 16	00	
№№ 17	C5	*PUSH B*
№№ 18	01	*LXI B*, 0018
№№ 19	18	
№№ 1A	00	
№№ 1B	C5	*PUSH B*
№№ 1C	3E	*MVI A*, 24
№№ 1D	24	
№№ 1E	C1	*POP B*
№№ 1F	81	*ADD C*
№№ 20	C1	*POP B*
№№ 21	91	*SUB C*
№№ 22	C1	*POP B*
№№ 23	81	*ADD C*
№№ 24	C1	*POP B*
№№ 25	81	*ADD C*
№№ 26	C1	*POP B*
№№ 27	91	*SUB C*
№№ 28	C1	*POP B*
№№ 29	91	*SUB C*
№№ 2A	C1	*POP B*
№№ 2B	81	*ADD C*
№№ 2C	76	*HLT*

Следует запомнить, что при записи данных адреса стека растут от больших к меньшим и указатель стека *SP*всегда содержит последний адрес стека, в котором записано число.

Память микроЭВМ, построенной на основе МПК серии К 580, может иметь не более 65536 однобайтовых ячеек. Учитывая ограниченные возможности памяти при разработке программ, нужно стараться сделать их как можно короче. С этой целью часть программы, которая неоднократно повторяется, или программа, которая часто используется, может быть оформлена в виде подпрограмм – последовательностей команд, выполнение которых может быть вызвано из любого места программы любое количество раз. Процесс передачи управления к подпрограмме называется её вызовом. Данные и адреса, требуемые для работы подпрограммы, передаваемые по окончании ее работы в основную программу, называются выходными параметрами.

Для вызова программы и возврата из нее используются команды – трехбайтовая *CALL*/A_2//A_1/ и однобайтовая *RET*(от англ. RETURN - возврат). Команда *CALL*/A_2//A_1/ загружает в программный счетчик микропроцессора

содержимое байтов /A_2/ /A_1/, записанных в последующих двух адресах памяти после адреса, по которому записан код команды *CALL/CD/*.Содержимое байта /A_2/ записывается в младший байт *PCL*программного счетчика, а третий байт /A_1/ команды – в старший байт *PCH*программного счетчика, при этом микропроцессор автоматически сохраняет в стеке адрес основной программы, ккоторому она будет обращаться после выполнения подпрограммы. Любая подпрограмма должна кончаться командой *RET*. Эта команда помещает в программный счетчик последнее записанное на данный момент в стеке число. Далее выполнение программы будет осуществляться с этого адреса.

Таким образом, вызов одной подпрограммы и возврат из нее осуществляется сравнительно простыми средствами: необходим 16-битный регистр для временного заполнения адреса возврата. В качестве такого регистра часто используется указатель стека. Команда *CALL*фактически выполняет функции команды *PUSHPC*.

Рассмотрим пример использования подпрограммы, которая извлекает из памяти ЭВМ два числа, складывает их в результат сложения запоминает по заданному адресу.

Подпрограмма

№№ 00	7E	*MOVA, M*
№№ 01	23	*INXH*
№№ 02	46	*MOVB, M*
№№ 03	80	*ADDB*
№№ 04	32	*STANN*10
№№ 05	10	
№№ 06	№№	
№№ 07	C9	*RET*

Подпрограмма заканчивается командой безусловного перехода к той команде основной программы, на которой ее прервали.

Программа

№№ 20	A1	
№№ 21	49	
№№ 30	21	*LXIH, NN*20
№№ 31	20	
№№ 32	№№	
№№ 33	CD	*CALLNN*00
№№ 34	00	
№№ 35	№№	
№№ 36	76	*HLT*

В этой программе в память микроЭВМ по адресам №№ 20 и №№ 21 записываются числа A1 и 49 соответственно. Затем, набрав код 21, загружается

регистровая пара *HL*, указывая начальный адрес памяти, где записаны числа. Необходимость в этой команде объясняется тем, что в рассмотренной подпрограмме команды пересылки *MOVA, МиMOVB, M* используют косвенную адресацию (т.е. через регистры *НиL*). После этих операций вызывается подпрограмма. При ее вызове указывается адрес входа - №№ 00, по которому передается управление. Одновременно с этим в стеке запоминается адрес №№ 35, по которому управление вновь будет возвращено в основную программу после окончания подпрограммы.

Ввести в память микроЭВМ и выполнить в непрерывном режиме программу и подпрограмму следующего вычисления:

02 * 15 + 60 / 12

Программа

№№ 50	06	*MVID, 02*
№№ 15	02	
№№ 52	1E	*MVIE, 15*
№№ 53	15	
№№ 54	CD	*CALLNN00*
№№ 55	00	
№№ 56	№№	
№№ 57	26	*MVIH, B*
№№ 58	2E	*MVI L, C*
№№ 59	22	*SHLD NN80*
№№ 5A	80	
№№ 5B	№№	
№№ 5C	16	*MVI D, 12*
№№ 5D	12	
№№ 5E	1E	*MVI E, 60*
№№ 5F	60	
№№ 60	CD	*CALL NN20*
№№ 61	20	
№№ 62	№№	
№№ 63	3A	*LDA NN81*
№№ 64	81	
№№ 65	№№	
№№ 66	84	*ADDH*
№№ 67	32	*STANN83*
№№ 68	83	
№№ 69	№№	
№№ 6A	E7	*RST 4*

ПОДПРОГРАММА УМНОЖЕНИЯ

Начальный адрес подпрограммы – №№ 00; выходные параметры: регистр D– множимое, регистр E – множитель. Результат перемножения записываются в регистры B, C.

№№ 00	01	*LXIB, 0000*	Очистить содержимое регистров BC
№№ 01		00	
№№ 02		00	
№№ 03	3E	*MVIA, 01*	Загрузить в аккумулятор указатель разряда
№№ 04		01	
№№ 05	A7	*ANAA*	Очистить фланг C
№№ 06	F5	*PUSHPSW*	Сохранить указатель разряда в стеке
№№ 07	A3	*ANAE*	Проверить содержимое очередного разряда множителя
№№ 08	78	*MOVA, B*	Загрузить в A старший байт суммы
№№ 09	CA	*IZNN00*	Если в очередном разряде записан 0, идти на адрес №№ 0D
№№ 0A		0D	
№№ 0B		№№	
№№ 0C	82	*ADDD*	Прибавить множимое к сумме
№№ 0D	1F	*RAR*	Сдвинуть сумму вправо (младший бит - C)
№№ 0E	47	*MOVB, A*	Сохранить содержимое A в регистре B
№№ 0F	79	*MOVA, C*	Загрузить в A младший байт суммы
№№ 10	1F	*RAR*	Сдвинуть число в A вправо (старший бит - C)
№№ 11	4F	*MOVC, A*	Сохранить содержимое A в регистре C
№№ 12	F1	*POPPSW*	Получить из стека указатель разряда
№№ 13	17	*RAL*	Указатель на следующий разряд
№№ 14	D2	*INCNN 06*	Если разряд не последний, продолжать по адресу №№ 06
№№ 15		06	
№№ 16		№№	
№№ 17	09	*RET*	Умножение закончено, возврат

ПОДПРОГРАММА ДЕЛЕНИЯ

Деление организуется процедурой, похожей на умножение. Отличие заключается в том, что если умножение выполняется по правилу «сложение – сдвиг», то деление выполняется по правилу «вычитание – сдвиг». Деление

основывается на последовательном вычитании делителя из делимого и остатков от деления. Входные параметры подпрограммы – делимое (в регистре E), делитель (в регистре D); выходные параметры – частное (в регистре H) и остаток (в регистре C).

№№ 20	21	*LXIH, 0008*	Загрузить счетчик битов (регистр L)и очистить регистр частного H
№№ 21		08	
№№ 22		00	
№№ 23	0E	*MVIC, 00*	Очистить регистр промежуточного деления
№№ 24		00	
№№ 25	7B	*MOVA, E*	Загрузить делимое в аккумулятор
№№ 26	17	*RAL*	Загрузить старший бит в разряд C
№№ 27	5F	*MOVE, A*	Возвратить делимое в регистр E
№№ 28	79	*MOVA, C*	Загрузить в A промежуточное делимое из регистра C
№№ 29	17	*RAL*	Сдвинуть разряд C в младший бит
№№ 2A	92	*SUBD*	Вычесть из содержимого A делитель
№№ 2B	D2	*INCNN2F*	Если C=1, восстановить содержимое аккумулятора
№№ 2C		0F	
№№ 2D		№№	
№№ 2E	82	*ADDD*	Сложить содержимое регистров A
№№ 2F	4F	*MOVC, A*	Возвратить промежуточное данное в регистр C
№№ 30	3F	*CMC*	Инвертировать разряд C
№№ 31	7C	*MOVA, H*	Сдвинуть разряд C в младший бит регистра частного H
№№ 32	17	*RAL*	
№№ 33	67	*MOVH, A*	
№№ 34	2D	*DCRL*	Проверены все восемь разрядов
№№ 35	C2	*INZNN25*	Если нет – продолжать
№№36		25	
№№37		№№	
№№38	C9	*RET*	Если да, деление закончено, возврат

Две первые команды инициализации сбрасывают регистры частного H и остатков C, а также загружают в регистр-счетчик L– длину частного. Затем делимое сдвигается на один бит влево, старший бит делимого передается во флажок C, а остальные биты возвращаются в регистр E. Остаток из регистра C передается в аккумулятор, а бит делимого сдвигается из флажка C в младший бит аккумулятора. С помощью команды *SUB*из остатка вычитается делитель.

Если результат положительный (заема нет и флажок С сброшен), то происходит переход по адресу №№ 2F. Если при вычитании возникает заем, то с помощью команды *ADD*восстанавливается положительный остаток. После этого производится инвертирования флажка С и образованная в нем цифра поступает в младший бит регистра частного Н. Наконец, команды *DCR*и*INZ* обеспечивают зацикливание для получения всех 8 бит частного.

ЗАДАНИЕ К ПРАКТИЧЕСКОЙ РАБОТЕ

Проанализировать программы и подпрограммы, приведенные в инструкции. Разработать несложную программу вычислений, когда под стековую память отводится произвольную область ОЗУ. Составить программу вычислений, включающую вызов подпрограмм сложения и вычитания.

СОДЕРЖАНИЕ ОТЧЕТА

Отчет должен содержать название и цель работы, перечень изученных команд и комментарии, результаты исследований приведенных в инструкции программ, а также программы, разработанные в соответствии с заданием к лабораторной работе.