Тема 10. Склад команд арифметичних дій.

Команды сложения

Первыми рассмотрим команды арифметических действий. Они сведены в табл. 9.1 и содержат команды сложить, вычесть, инкрементировать, декрементировать, сравнить. Заметим по данным табл. 10.1, что существуют четыре команды сложения. Аккумулятор (регистр А) содержит одно из слагаемых. Каждая команда точно оговаривает различные источники другого слагаемого.

Рассмотрим первую из команд сложения в табл. 10.1. Команда сложить непосредственно является двухбайтовой. Ее формат КОП (в этом случае С6Н) содержится в первом байте команды, непосредственно за ним — во второмбайте — находятся данные для сложения с содержимым аккумулятора. Команда ADI выполняется по схеме, приведенной на рис. 10.1, *а*. Данные, находящиеся в памяти непосредственно за КОП, складываются с содержимым аккумулятора ($0000\ 1111_2$). Сумма ($0001\ 1111_2$) помещается в аккумулятор.

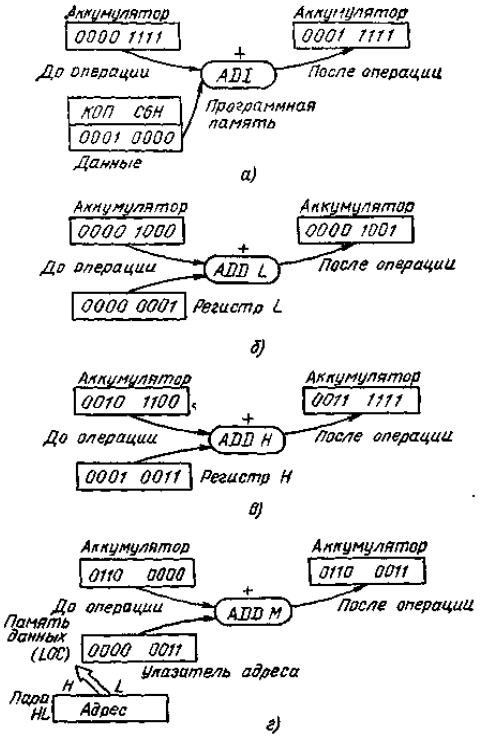


Рис. 10.1. Интерпретация операции ADD *M в* кодах табл. 10.1:
а—сложение (А) с данными, следующими непосредственно за КОП; *б* —сложение (L) и (Д); *в* — сложение (Н) и (А); *г* —сложение с косвенной адресацией

Таблица 10.1, Команды арифметических операций типового МП

Операция	Адресация	Мнемоника	К.ОП	Байты	формат команды	Символика	Индикаторы
СложитьА с данными	Непосредственная	AD1	C6	2	КОП данные	(A) ← (A) + (байт 2)	Z, CY
Сложить LcA	Регистровая	ADD L	85	1	КОП	(A) ← (A) + (L)	Z, CY
Сложить H с A	»	ADD H	84	1	коп	(A) ← (A) + (H)	Z, CY
Сложить LOC (HL) cA	Косвенная регистровая	ADD M	86	1	коп	(A) ← (A) + ((H)(L))	Z, CY
Вычесть данные изА	Непосредственная	SUI	6	2	КОП данные	(A) ← (A) - (байт 2)	Z, CY
Вычесть LизА	Регистровая	SUB L	95	1	коп	(A) ← (A) - (L)	Z, CY
Вычесть H изА	Регистровая	SUB H	94	1	коп	(A) ← (A) - (H)	Z, CY
Вычесть LOC (HL) изА	Косвенная регистровая	SUB M	96	1	коп	(A) ← (A) - ((H)(L))	Z, CY
ИнкрементА	Регистровая	INR A	3C	1	КОП	(A) ← (A) + 1	Z
Инкремент HL	»	INX H	23	1	КОП	(HL) ← (HL) + 1	
ДекрементА	»	DCR A	3D	1	КОП	(A) ← (A) - 1	Z
Декремент HL	»	DCX H	2B	1	КОП	(HL) ← (HL) - 1	
СравнитьА с данными	Непосредственная	CPI	FE	2	КОП данные	(A) ← (байт 2)	Z= 1, если (A) = (байт 2); CY =1, если (A) < (байт 2)
СравнитьА с L	| Регистровая	CMP L	BD	1	коп	(A) ← (L)	Z = l, если (A) = (L); CY = 1, если (A) < (L)
СравнитьА с H	»	CMP H		1	КОП	(A) ← (H)	Z = 1, если (A) = (H); CY= 1, если (A) < (H)
СравнитьА cLOC (HL)	Косвенная регистровая	CMP M		1	коп	(A) ← ((H)(L))	Z = l, если (A)=((H)(L)); CY=1,если (A)<((H)(L))

Второй является команда сложить содержимое регистров *L*и *A* (мнемоника ADD L), на рис. 10.1, *б* показано ее выполнение. Содержимое аккумулятора (0000 1000_2) складывается с содержимым регистра *L*(0000 0001_2), получающаяся в результате выполнения команды ADD *L*сумма (0000 1001_2) помещается в аккумулятор.

Третья команда в табл. 10.1 представляет собой однобайтовую команду сложить содержимое регистров *H* и*A* (мнемоника ADD *H).* Это другая команда сложения содержимого одного регистра с содержимым другого, и она выполняется в последовательности, приведенной на рис. 10.1, *в.* Содержимое регистра *A* (0010 1100_2) сложено с содержимым регистра *H* (00010011_2), сумма (0011 1111_2) помещена в аккумулятор.

Четвертая строка таблицы представляет собой однобайтовую команду сложить с косвенным регистром (мнемоника ADD *M).* Адрес данного слагаемого числа задан в более сложной форме с использованием способа косвенной регистровой адресации. Рисунок 10.1, *г* является примером выполнения команды ADD *M.* Пара *HL*указывает 16-разрядный адрес памяти, т. е. LOC. Содержимое LOC (0000 0011_2) сложено с содержимым аккумулятора (0110 0000_2), сумма (0110 0011_2) помещена в аккумулятор. Команды косвенного сложения используют в качестве указателя адреса 16-разрядный регистр (обычно пару *HL).*

Рассмотрим снова команду сложить с косвенным регистром (ADD M) в табл. 10.1. Предписание символики указывает: сложить LOC (H + L) с*A,* которое мы прочтем как сложить содержимое памяти, на которую указывает пара регистров *HL,*с содержимым регистра *A.* Следуя

соответствующей строке в табл. 10.1, мы найдем, что способом адресации является косвенная регистровая и соответствующей мнемоникой будет ADD *M* с КОП 86Н. Эта команда является однобайтовой. Представленный формат ее показывает, что единственным байтом является КОП. Следуя предписанию, данному фирмой Intel (и, очевидно, обратному обычной записи), символическую запись читают справа налево. Справа стрелки первый операнд идентифицирован как содержимое *A* (акумулятора). Скобки в рамках этой записи означают содержимое. Знак «+» означает выполняемую операцию, а двойная скобка (()) указывает команду косвенной адресации. Выражение((H)(L)) означает: содержимое пары *HL*указывает адрес расположения в памяти второго операнда. Иначе говоря, второй операнд расположен в ячейке памяти, на которую указывает пара регистров *HL*.После того как операция выполнена, ее результат помещается в аккумулятор. Все действия выполняются по такой схеме:

Результат	Первый операнд	Операция	Второй операнд
(A)←	——(A)	+	((H)(L))
Содержимое регистра *A*	Содержимое регистра *A*	Сложить	Содержимое пары указывает на положение операнда в памяти

В последней правой колонке табл. 10.1 приведен список устанавливаемых по результатам операции индикаторов. Пример сложения двоичных чисел 1111 1111 и 0000 0001 приведен на рис. 10.2. Выполнение такой операции обычным способом дает следующий результат:

```
  1111 1111
+
  0000 0001
-------------
1 0000 0000
```

Перенос Содержимое аккумулятора

Решение этой задачи микропроцессор выполняет в соответствии с рис.10.2 (операция ADD *M.*

Аккумулятор содержит 1111 1111_2, тогда как ячейка памяти, на которую указывает пара регистров *HL,*содержит другое слагаемое (0000 0001_2). Выполнив операцию сложения, аккумулятор содержит 8 младших бит суммы,т. е. 0000 0000_2. Индикатор переноса регистра состояния установлен в 1, что показывает наличие переноса старшего бита результата. Индикатор нуля проверяет содержимое аккумулятора после операции и находит 0000 0000_2. Так как это нуль, индикатор нуля становится 1, показывая, что содержимое аккумулятора 0000 0000_2 после операции сложения.

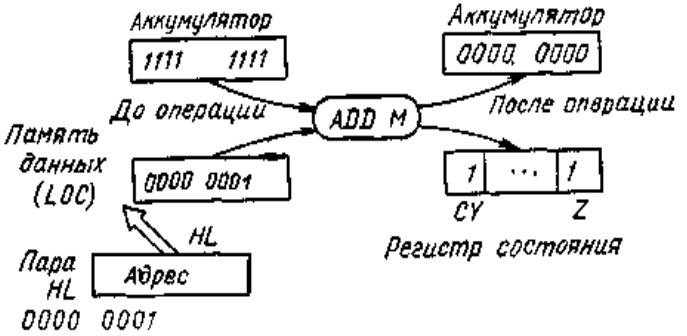


Рис. 10.2. Влияние результата операции ADD *M* на содержимое аккумулятора и индикаторы

Пример 1. Сложить два числа 10100011 и 00011110.

Далее приводится два варианта решения задачи. 1-й вариант составления программы:

Адрес	Ассемблерный код	Комментарий
00	*MVI A 10100011 B*	;Пересылка непосредственных
01		данных в *A*
02	*ADI A 00011110 B*	; Сложение непосредственных
03		данных с содержимым *A*
04	*END*	; Конец.

2-й вариант составления программы:

Адрес	Ассемблерный код	Комментарий
00	*MVI B 10100011 B*	;Переслать непосредственные
01		данные в регистр *B*
02	*MOV A, B*	; Переслать данные из *B* в *A*
03	*MVI B 10100011 B*	;Переслать непосредственные
04		данные в регистр *B*
05	*ADD B*	;Сложить содержимое A и *B*
06	*END*	;Конец.

Первый вариант по сравнение со вторым более предпочтителен, так как программа в первом случае занимает меньший объем памяти.

Пример 2 Сложить два числа 10100011 и 00011110 и результат записать в ячейку памяти 1101.

Программа:

Адрес	Ассемблерный код	Комментарий
00	*MVI B 10100011 B*	;Переслать непосредственные
01		данные в регистр B
02	*MOV A, B*	;Переслать содержимое *B* в *A*
03	*ADI B 10100011 B*	; Сложить непосредственные
04		данные с содержимым *A*
05	*IXI B 00000000*	;Загрузить пару регистров *BC*
06	*00001101B*	;непосредственными данными
07		
08	*STAX B*	; Записать содержимое *A* в ячейку памяти, адресуемую содержимым регистром *BC*

09 *END* ; Конец.

Пример 3 Сложить два числа 10100011 и 00011110 и результат записать по адресу, находящемуся в двойном регистре *HL* .

Программа:

Адрес	Ассемблерный код	Комментарий
00	*MVI 10100011 B*	;Пересылка непосредственных
01		данных в *A*
02	*ADI A 00011110 B*	;Сложение непосредственных
03		данных с содержимым *A*
04	*HCHG*	; Обмен содержимым регистров *H* и *D* , *E* и *L*.
05	*STAX D*	; Запись содержимого *A* в ячейку памяти, адресуемую содержимым регистров D и E
06	*END*	; Конец.

Пример 4 Сложить содержимое ячеек 40*H* и 41*H*. Результат поместить в ячейку 42*H*.

Программа

Адрес	Ассемблерный код	Комментарий
00	*LDA 40H*	; Загрузка *A* содержимым
01		; ячейки *40H*
02		;
03	*MOV B, A*	; Пересылка в регистр *B* содержимого *A*
04	*LDA 41H*	; Загрузка *A* содержимым
05		; ячейки *41H*
06		
07	*ADD B*	; Сложение содержимого *B* с содержимым *A*
08	*STA 42H*	; Пересылка содержимого *A* в
09		; ячейку *42 H*
0A		;
0B	*END*	; Конец.

Пример 5 Сложить число *1EH* с числами, находящимися в ячейках памяти с адресами 2000 *H* и 2005 *H* .

Программа:

Адрес	Ассемблерный код	Комментарий
00	*LXI H, 2000H*	; Установить адрес 2000*H*
01		; в *HL*
02		;
03	*MOV A, M*	;Переслать содержимое ячейки памяти, адрес которой находится в *HL* , в *A*
04	*ADI 1EH*	;Сложить число, содержащееся
05		; в *A*, с числом *1EH*
06	*LXI H,2005H*	; Загрузить адрес 2005 *H*
07		; в *HL*
08		;
09	*MOV B, M*	; Переслать число из ячейки памяти с адресом, находящимся в *HL* , в *B* .
0A	*ADD B*	; Сложить содержимое *A* с числом, находящимся в *B*
0B	*LXI H REZ*	; Установить в *HL* адрес,
0C		; определяемый меткой *REZ.*
0D		;
0E	*MOV M, A*	; Переслать результат из *A* в ячейку памяти с адресом в *HL*
0F	*REZ: DS 0IH*	; Зарезервировать 1 байт памяти для хранения результата
10	*END*	; Конец.

Команды вычитания

Рассмотрим теперь четыре операции вычитания, приведенные в табл. 10.1, относящиеся к составу команд типового микропроцессора. Каждая команда вычитает содержимое некоторого регистра или ячейки памяти из содержимого аккумулятора. В силу внутренних особенностей АЛУ не обладает возможностями

вычитания, оно осуществляет сложение, представляя вычитаемое в форме дополнительного кода и затем складывая его.

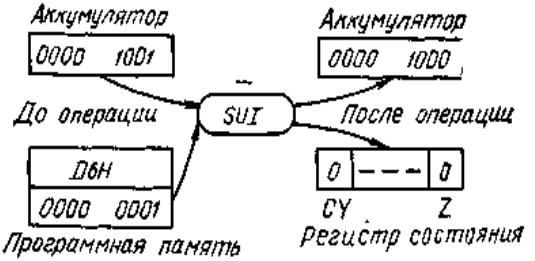
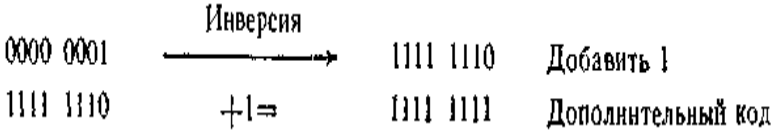


Рис. 10.3. Команда вычитания данных с непосредственной адресацией

В качестве примера рассмотрим процесс вычитания, представленный на рис. 10.3. Двоичное число 0000 0001 вычитается из 0000 1001 (09H — 01H = 08H). Выполнение этой операции обычным способом дает следующий результат:

CY				
	Инверсия	+	0000 1001 1111 1111	Дополнительный код
0	←	1	0000 1000	
Нет переноса		Перепол-нение	Содержимое аккумулятора (8 бит)	

Напомним, что вычитание двоичных чисел осуществляют сложением первого числа со вторым, представленным в форме дополнительного кода, пренебрегая переполнением. В этой задаче вторым числом будет 0000 0001, которое будет преобразовано в дополнительный код и даст:

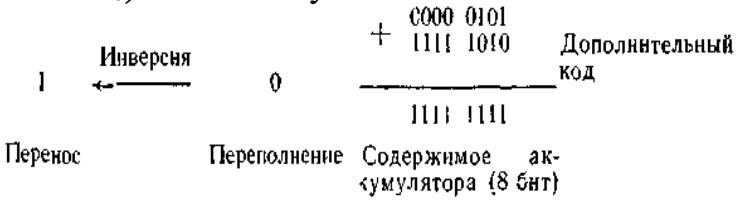


Дополнительный код 1111 1111 другого числа складывается тогда с первым числом, что дает сумму 1 0000 1000. В старшем бите суммы 1 является переполнением и не принадлежит разности 0000 1000_2. Микропроцессор использует это переполнение для установления индикатора переноса. Вычитая, МП инвертирует переполнение, и результат становится содержимым индикатора переноса *CY*. Переполнение 1 инвертируется и сбрасывает индикатор переноса в 0. Когда в ходе вычитания индикатор переноса сбрасывается, это значит, что переноса не было и что первое число больше второго.

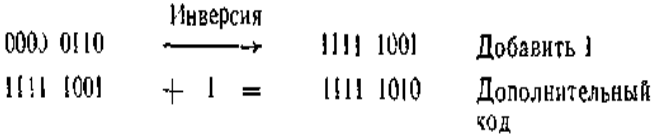
Команда вычесть непосредственные данные (мнемоника SUI) используется в примере на рис. 10.3. Следующие непосредственно заКОП данные второго байта памяти 0000 0001_2 вычитаются из содержимого аккумулятора 0000 1001_2, разность передается в аккумулятор, после чего операция завершается. Индикатор переноса сбрасывается в 0. Это означает, что переноса не было или содержащееся в аккумуляторе число до операции было больше числа в памяти. Индикатор нуля проверяет содержимое аккумулятора. Операция вычитания завершается, содержимое 0000 1000— не нуль, индикатор нуля также сброшен в 0.

Рассмотрим другой пример вычитания, когда уменьшаемое меньше вычитаемого. Пусть надо вычесть

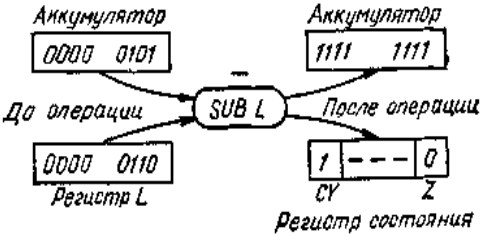
двоичное число 0000 0110 из двоичного 0000 0101 (05H—06H = FFH = -1_{10}). В этом случае:



Вычитаемое здесь представлено в дополнительном коде



Уменьшаемое 0000 0101_2 сложено с результатом преобразования вычитаемого в дополнительный код 1111 1010, что дает 1111 1111. Как можно увидеть, результат 1111 1111 является представлением в дополнительном коде числа -1_{10}. Сложение не вызывает переполнения тогда, когда имеем 0 в регистре переполнения. Этот результат инвертируется (так как речь идет о вычитании), что дает 1 в регистре переноса *CY*. Когда индикатор *CY* устанавливается в 1 после вычитания, это означает, что число, содержащееся в аккумуляторе, младше числа в памяти или в регистре. Индикатор переноса в состоянии 1 показывает, что число, содержащееся в аккумуляторе после того, как вычитание было выполнено, является дополнительным кодом, представляя отрицательное число. 1111 1111 в аккумуляторе на рис. 10.4 представляет собой -1_{10}.



Рнс. 10.4. Команда вычитания (L) из (A)

Пример 1 Из числа *A3H* вычесть число *1EH* . Результат занести в ячейку памяти с адресом *1030H.*

Программа:

Адрес	Ассемблерный код	Комментарий
00	*MVI A, 043H*	; Загрузить в *A* число *OA3H*
01		
02	*SUI I E H*	; Вычесть из числа *OA3H*
03		; число *IEH*
04	*LXI H, 1030H*	;Установить адрес *I030H*
05		в *HL*
06		
07	*MOV M, A*	; Переслать результат из *A* в ячейку памяти, адресуемую парой *HL*
08	*END*	; Конец.

Таким образом, под управлением команд рассмотренной группы микропроцессор может выполнять в АЛУ арифметические операции. Поскольку МП является одноадресным, то один из операндов в арифметических операциях всегда помещается в аккумулятор, неявно адресуемый самим кодом операции. Вслед за кодом операции необходимо указывать местонахождение второго операнда. Результат (сумма или разность) помещается в аккумулятор.

Операнд, который перед арифметической операцией находился в аккумуляторе, после выполнения операции уничтожается.

Команда сложения ADD, ADC

ADD r; A+B→A

ADC r; A+B+CY→A

Команда ADD обеспечивает сложение содержимого аккумулятора с содержимым другого регистра, при этом результат операции остаётся в аккумуляторе.

Команда ADC является разновидностью команды ADD. По команде ADC происходит не только сложение двух операндов, но и сложение с признаком переноса CY, оставшимся от предыдущей операции. Результат сохраняется в аккумуляторе.

Команда сложения ADI, ACI

Команда ADI осуществляет сложение операнда, который непосредственно следует за кодом операции, с содержимым аккумулятора. По команде ACI непосредственный операнд суммируется с содержимым аккумулятора и с признаком переноса CY

ADI data8; A+ data8 →A

ACI data8; A+ data8 +CY→A

Команда вычитания SUB, SBB

Команда SUB позволяет микропроцессору непосредственно вычесть содержимое одного из регистров общего назначения или ячейки памяти M из содержимого аккумулятора. Команда SBB является разновидностью команды SUB. По этой команде осуществляется вычитание с заемом.

SUB r A – r →A

SUB M A – M →A

SBB r A – r - CY →A

SBB M A – M - CY →A

Команда вычитания SUI, SBI

По команде SUI из содержимого аккумулятора вычитается операнд, который непосредственно следует за кодом операции. По команде SBI из содержимого аккумулятора вычитается и непосредственный операнд, и признак заёма CY.

SUI data8; A+ data8 →A

SBI data8; A+ data8 +CY→A

Внимание! Все описанные арифметические команды сложения и вычитания изменяют (модифицируют) содержимое всех признаков регистра признаков.

Команды INR, DCR

Команда INR является разновидностью команды ADD. По этой команде МП увеличивает на 1 содержимое одного из регистров РОН, аккумулятора или ячейки памяти М.

Команда DCR является разновидностью команды вычитания SUB. По этой команде МП уменьшает на 1 содержимое одного из регистров РОН, аккумулятора или ячейки памяти М.

INR r r + 1 → r

INR M M + 1 → M

DCR r r – 1 → r

DCR M M – 1 → M

Внимание! Эти команды модифицируют все признаки за исключением признака переноса CY.

Команды INX, DCX

Команда инкремента INX и декремента DCX позволяют соответственно увеличить и уменьшить на 1 содержимое регистровых пар (B-, D- , H- пары) и указателя стека SP.

INX rp; rp + 1 → r

DCX rp; rp – 1 → r

Внимание! Эти команды не модифицируют регистр признаков.

Команды DAD, DAA

Команда двойного сложения DAD суммирует содержимое регистровой пары (H,L) и адресуемой регистровой пары rp (rp это B-,D- H- пары, SP)

DAD rp; (H,L) + rp → (H,L)

Внимание! Эта команда модифицирует только признак переноса.

Команда десятичной коррекции аккумулятора DAA осуществляет перевод 8- разрядного двоичного числа в аккумуляторе в две цифры двоично-десятичного кода с правильной установкой признака переноса CY. При этом производятся следующие действия: 1. Если младшая тетрада содержит число, больше 910, или установлен признак вспомогательного переноса AC=1, то содержимое аккумулятора увеличивается на 610. 2. Если после этого старшая тетрада аккумулятора содержит число, большее 9, или установлен признак вспомогательного переноса CY=1, то в старшую тетраду прибавляется 610

Внимание! При десятичной коррекции модифицируются все признаки регистра признаков.

Кроме рассмотренных типовой микропроцессор

обладает четырьмя командами сравнения (см. четыре последние строки в табл. 9.1). Команды сравнения вычитают содержимое регистра или ячейки памяти из содержимого аккумулятора, но не изменяют содержимое ни того, ни другого. Индикаторы же подвержены воздействию команд сравнения.

На рис. 10.5 приведен пример использования команды СРАВНИТЬ регістр *L*.

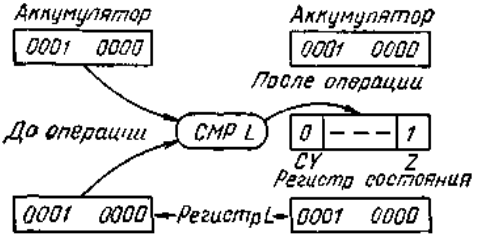
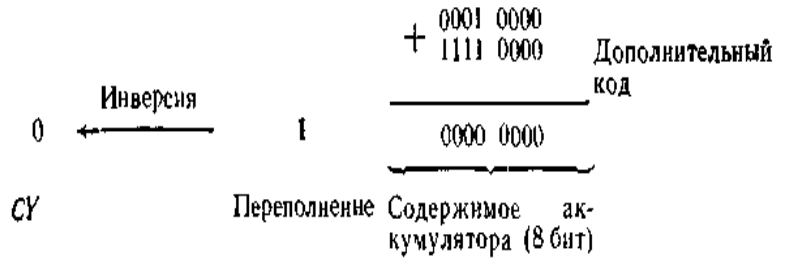


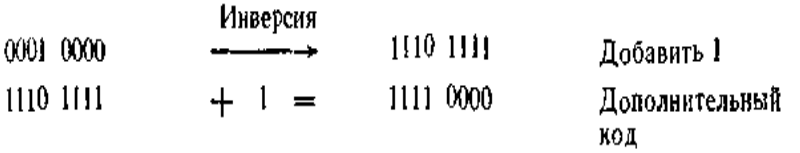
Рис. 10.5. Команда сравнения (A) с (L)

Равные числа $0001\ 0000_2$ являются содержимым аккумулятора и регистра *L*и сравниваются микропроцессором. Заметим, что ни одно, ни другое из содержимых не изменяется после операции сравнения. Индикаторы подвержены влиянию результата сравнения.

Согласно приведенному в табл. 10.1 символическому представлению информации содержимое регистра *L* вычтено из содержимого акумулятора *A* по команде CMP *L:*



Второе число (содержимое регистра L) переведено в дополнительный код:



Первое число 0001 0000_2 и дополнительный код второго числа 1111 0000 складывается, что дает 1 0000 0000. Затем проверяется равенство 0 восьми младших разрядов. Индикатор нуля принимает значение 1. Переполнение 1 инвертируется АЛУ, и в этом примере индикатор переноса принимает значение 0. Сброшенный индикатор переноса *CY* означает, что содержимое аккумулятора больше или равно содержимому регистра *L*.