**Система команд микроконтроллеров семейства MCS-48 (1816ВЕ48)**

Система команд микроконтроллера 1816ВЕ48 предоставляет большие возможности обработки данных, обеспечивает реализацию логических, арифметических операций, а также управление в режиме реального времени.

Реализована побайтовая (8 бит) и потетрадная (4 бита) обработка данных. Микросхемы семейства MCS-48 — это 8-разрядные микропроцессоры, а это означает, что ПЗУ, ОЗУ, регистры специального назначения, [АЛУ](http://digteh.ru/CVT/alu.php) и внешние шины имеют байтовую организацию.

В машинном коде команда занимает один, или два байта в зависимости от типа команды. Команды выполняются за один или два машинных цикла. При этом большинство команд выполняются за один машинный цикл длительностью 2,5 мкс (при 6 МГц кварцевом резонаторе).

Запись команд в машинных кодах для человека неудобна, кроме того, разные машинные команды выполняют одинаковые действия только над разными ячейками памяти. Поэтому для записи команд микропроцессоров была придумана система мнемонических обозначений. Для записи команды микропроцессора сначала ставится мнемоническое обозначение, затем указывается ячейка памяти – приёмник результата выполнения операции и наконец источник данных для выполнения операции. Например, в команде

MOV A, R2

символы MOV обозначают операцию копирования, второй операнд R2 определяет, что данные необходимо взять из регистра R2 текущего банка регистров, а первый операнд A определяет, что результат необходимо поместить в регистр – аккумулятор. При этом старое значение регистра – аккумулятора будет стёрто.

Мнемоническое обозначение команды отделяется от операндов одним или несколькими символами пробела или табуляции, а операнды отделяются друг от друга запятыми.

Если операция требует для выполнения двух источников и одного приёмника результата операции (например, команда сложения ADD или логического умножения ANL), то первый операнд является одновременно и источником и приёмником результата операции. Например, в команде

ADD A, R1

символы ADD обозначают операцию сложения двух чисел, данные будут взяты из регистра R1 текущего банка регистров и аккумулятора а результат будет помещён в аккумулятор вместо старого значения этого регистра.

В таблице 1 приведены инструкции, влияющие на установку флагов.

Таблица 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Мнемоника** | **Флаги** | | | | | | |
| **C** | **AC** | **F0** | **F1** | **TF** | **BS** | **DBF** |
| ADD, ADDC | + | + | - | - | - | - | - |
| DA | + | + | - | - | - | - | - |
| CLR C | 0 | - | - | - | - | - | - |
| CPL C | + | - | - | - | - | - | - |
| CLR F0 | - | - | 0 | - | - | - | - |
| CPL F0 | - | - | + | - | - | - | - |
| CLR F1 | - | - | - | 0 | - | - | - |
| CPL F1 | - | - | - | + | - | - | - |
| JTF | - | - | - | - | 0 | - | - |
| RRC | + | - | - | - | - | - | - |
| RLC | + | - | - | - | - | - | - |
| MOV A, PSW | + | + | + | - | - | + | - |
| RETR | + | + | + | - | - | + | - |
| SEL MB0, SEL MB1 | - | - | - | - | - | - | + |
| SEL RB0, SEL RB1 | - | - | - | - | - | + | - |

Систему команд микроконтроллера условно можно разбить на пять групп:

* [арифметические команды](http://digteh.ru/MCS48/SysInstr.php#ArifmInstr)
* [логические команды](http://digteh.ru/MCS48/SysInstr.php#CtrlInstr)
* [команды передачи данных](http://digteh.ru/MCS48/SysInstr.php#MovInstr)
* [команды ветвления программ и передачи управления](http://digteh.ru/MCS48/SysInstr.php#UprInstr)
* [команды управления режимами работы микроконтроллера](http://digteh.ru/MCS48/SysInstr.php#CtrlInstr)

**Арифметические команды**

В наборе команд микроконтроллера имеются следующие арифметические операции:

* [сложение ADD](http://digteh.ru/MCS48/instr1.php#ADD),
* [сложение с учетом флага переноса ADDC](http://digteh.ru/MCS48/instr1.php#ADDC),
* [инкрементирование (увеличение на 1) INC,](http://digteh.ru/MCS48/instr1.php#INC)
* [декрементирование (уменьшение на 1) DEC](http://digteh.ru/MCS48/instr1.php#DEC),
* [десятичная коррекция DA](http://digteh.ru/MCS48/instr1.php#DA),

Действия производятся над целыми числами без знака. Команды сложения ADD позволяют сформировать перенос в следующий разряд. Команды сложения с учётом знака переноса позволяют выполнять многобайтное сложение. Вычитание может быть реализовано переводом одного из операндов в дополнительный код с последующим выполнением операции суммирования. Более сложные операции такие как умножение или деление могут быть реализованы только программным способом.

**Логические команды**

Система команд микроконтроллера позволяет реализовать логические операции:

* И ( [ANL](http://digteh.ru/MCS48/instr1.php" \l "ANL) [ANLD](http://digteh.ru/MCS48/instr1.php#ANLD)),
* ИЛИ ( [ORL](http://digteh.ru/MCS48/instr2.php" \l "ORL) [ORLD](http://digteh.ru/MCS48/instr2.php#ORLD)),
* ИСКЛЮЧАЮЩЕЕ ИЛИ ( [XRL](http://digteh.ru/MCS48/instr2.php" \l "XRL)).
* сброс и инвертирование всех восьми разрядов аккумулятора ( [CPL A](http://digteh.ru/MCS48/instr1.php" \l "CPL_A); [CLR A)](http://digteh.ru/MCS48/instr1.php#CLR_A);
* сброс и инвертирование флагов микроконтроллера ( [CLR C](http://digteh.ru/MCS48/instr1.php" \l "CLRC); [CLR F0](http://digteh.ru/MCS48/instr1.php#CLRF0); [CLR F1](http://digteh.ru/MCS48/instr1.php#CLRF1); [CPL C](http://digteh.ru/MCS48/instr1.php#CPL_C); [CPL F0](http://digteh.ru/MCS48/instr1.php#CPL_F0); [CPL F1](http://digteh.ru/MCS48/instr1.php#CPL_F1));
* циклический сдвиг влево и вправо с учетом переноса и без ( [RR A](http://digteh.ru/MCS48/instr2.php" \l "RR); [RRC A](http://digteh.ru/MCS48/instr2.php#RRC); [RL A](http://digteh.ru/MCS48/instr2.php#RL); [RLC A](http://digteh.ru/MCS48/instr2.php#RLC));
* обмен местами старшей и младшей тетрад внутри аккумулятора ( [SWAP A](http://digteh.ru/MCS48/instr2.php" \l "SWAP)).

**Команды пересылки данных**

Как было рассмотрено ранее, арифметические и логические команды могут быть выполнены только над содержимым регистра аккумулятора, поэтому исключительно важное значение в системе команд приобретают команды пересылки данных. С помощью этих команд можно скопировать содержимое любой ячейки памяти в регистр-аккумулятор или наоборот скопировать содержимое аккумулятора в любую ячейку памяти. Так как в микроконтроллере присутствует три независимых области памяти, то для обращения к ним введены различные команды:

* обмен данными во внутреннем ОЗУ: [MOV](http://digteh.ru/MCS48/instr2.php#MOV), XCH
* обмен данными из внешней памяти данных: [MOVX](http://digteh.ru/MCS48/instr2.php#MOVX)
* копирование данных из памяти программ: [MOVP](http://digteh.ru/MCS48/instr2.php#MOVP)
* обмен данными параллельных портов [IN](http://digteh.ru/MCS48/instr1.php#IN), [OUTL](http://digteh.ru/MCS48/instr2.php#MOV), [MOVD](http://digteh.ru/MCS48/instr2.php#MOVD)

Примеры использования команд пересылки данных:

Любая ячейка 256- байтового блока внутреннего ОЗУ данных может быть выбрана с использованием косвенно-регистровой адресации через регистры указатели R0 и R1 (выбранного банка рабочих регистров):

[MOV A, @R0](http://digteh.ru/MCS48/instr2.php#MOV) ;Скопировать число из ячейки памяти с адресом, хранящемся в R0, в аккумулятор

[MOV @R1, A](http://digteh.ru/MCS48/instr2.php#MOV) ;Скопировать число из аккумулятора, в ячейку памяти с адресом, хранящемся в R1

Таблицы символов (кодов), записанные в ПЗУ программы могут быть скопированы в аккумулятор с помощью команд передачи данных с [косвенной адресацией](http://digteh.ru/MCS48/SysInstr.php#Indir_Reg_Adr):

[MOVP A, @A](http://digteh.ru/MCS48/instr2.php#MOVP) ;Скопировать символ из текущей страницы памяти программ в аккумулятор

[MOVP3 A, @A](http://digteh.ru/MCS48/instr2.php#MOVP3) ;Скопировать символ из третьей страницы памяти программ в аккумулятор

Ячейка адресного пространства 256 байт внешнего ОЗУ также может быть выбрана с использованием [косвенно-регистровой](http://digteh.ru/MCS48/SysInstr.php#Indir_Reg_Adr) адресации через регистры R0 или R1:

[MOVX A, @R0](http://digteh.ru/MCS48/instr2.php#MOVX) ;Скопировать число из внешней ячейки памяти с адресом, ;хранящемся в R0, в аккумулятор

[MOVX @R1, A](http://digteh.ru/MCS48/instr2.php#MOV) ;Скопировать число из аккумулятора, в ячейку памяти с адресом, хранящемся в R1

Содержимое аккумулятора может быть обменено с содержимым рабочих регистров выбранного банка:

XCH A, R0.

Для работы с параллельными портами используется отдельные команды пересылки данных IN и OUTL:

[IN A, P0](http://digteh.ru/MCS48/instr1.php#IN) ;Скопировать число из порта P0, в аккумулятор

[OUTL P1, A](http://digteh.ru/MCS48/instr2.php#MOV) ;Скопировать число из аккумулятора, в порт P1

**Команды ветвления и передачи управления**

Команды ветвления позволяют реализовывать условные операторы и операторы циклов. В микроконтроллерах семейства MCS-48 доступны следующие команды:

* безусловный переход: [JMP](http://digteh.ru/MCS48/instr1.php#JMP)
* Вызов и возврат из подпрограммы: [CALL](http://digteh.ru/MCS48/instr1.php#CALL), [RET](http://digteh.ru/MCS48/instr2.php#RET), RETR
* проверка содержимого аккумулятора: [JZ](http://digteh.ru/MCS48/instr2.php#JZ), [JNZ](http://digteh.ru/MCS48/instr2.php#JNZ), [JMPP](http://digteh.ru/MCS48/instr1.php#JMPP), [JB0](http://digteh.ru/MCS48/instr1.php#JB0) ... [JB7](http://digteh.ru/MCS48/instr1.php#JB7)
* проверка флага переноса С: [JC](http://digteh.ru/MCS48/instr1.php#JC), [JNC](http://digteh.ru/MCS48/instr1.php#JNC)
* проверка сигнала на ножке T0: [JT0](http://digteh.ru/MCS48/instr2.php#JT0), [JNT0](http://digteh.ru/MCS48/instr2.php#JNT0)
* проверка сигнала на ножке T1:  [JT1](http://digteh.ru/MCS48/instr2.php#JT1), [JNT1](http://digteh.ru/MCS48/instr2.php#JNT1)
* проверка сигнала на ножке INT: [JNI](http://digteh.ru/MCS48/instr2.php#JNI)
* Проверка состояния флагов пользователя [JF0](http://digteh.ru/MCS48/instr1.php#JF0), [JF1](http://digteh.ru/MCS48/instr1.php#JF1)

**Описание машинных команд микроконтроллеров семейства MCS-48 (1816ВЕ48)**

**ADD A, <байт-источник>** - сложение.

Описание: складывает содержимое аккумулятора А с содержимым байта-источника. Результат операции помещается в аккумулятор. Флаг переноса C устанавливается при переносе из 7го разряда. Флаг вспомогательного переноса AC устанавливается при переносе из 3го разряда. При сложении знаковых чисел флаг переполнения OV устанавливается при превышении максимального положительного числа или при получении числа меньшего минимально возможного. Для команды сложения разрешены следующие режимы адресации байта - источника:

* [регистровая](http://digteh.ru/MCS48/SysInstr.php#Reg_Adr);

**Пример:**ADD A, R7

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (A) + (Rn) , n=0,1,2,3,4,5,6,7  
**Машинный код:**

0 1 1 0 1 r r r

* [косвенно-регистровая](http://digteh.ru/MCS48/SysInstr.php#Indir_Reg_Adr);

**Пример:**ADD A, @R0

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (A) + ((Ri)) , i=0,1  
**Машинный код:**

0 1 1 0 0 0 0 i

* [непосредственная](http://digteh.ru/MCS48/SysInstr.php#Imm_Adr).

**Пример:**ADD A, #38h

**Количество байт: 2  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 2  
(A) = (A) + (date 8)  
**Машинный код:**

0 0 0 0 0 0 1 1

i i i i i i i i

**ADDC A, <байт-источник>** - сложение с учетом переноса.

Описание: складывает содержимое аккумулятора А с содержимым байта-источника, к получившемуся результату прибавляется значение флага переноса. Результат операции помещается в аккумулятор. Флаг переноса C устанавливается при переносе из 7го разряда. Флаг вспомогательного переноса AC устанавливается при переносе из 3го разряда. При сложении знаковых чисел флаг переполнения OV устанавливается при превышении максимального положительного числа или при получении числа меньшего минимально возможного. Для команды сложения разрешены следующие режимы адресации байта - источника:

* [регистровая](http://digteh.ru/MCS48/SysInstr.php#Reg_Adr);

**Пример:**ADDC A, R3

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (A) + (C) + (Rn) , n=0, : ,7  
**Машинный код:**

0 1 1 1 1 r r r

* [косвенно-регистровая](http://digteh.ru/MCS48/SysInstr.php#Indir_Reg_Adr);

**Пример:**ADDC A, @R0

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (A) + (C) + ((Ri)) , i=0,1  
**Машинный код:**

0 1 1 1 0 0 0 i

* [непосредственная](http://digteh.ru/MCS48/SysInstr.php#Imm_Adr).

**Пример:**ADDC A, #100

**Количество байт: 2  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 2  
(A) = (A) + (C) + (date 8)  
**Машинный код:**

0 0 0 1 0 0 1 1

i i i i i i i i

**ANL <байт приемник>,<байт источник>** - команда "логическое И" для байтовых переменных.

**Описание**: выполняет операцию побитового "логического И" над указанными переменными и помещает результат в байт приемник. Команда не воздействует на флаги.

Для команды ANL с аккумулятором в качестве байта назначения разрешены следующие режимы адресации байта - источника:

* [регистровая](http://digteh.ru/MCS48/SysInstr.php#Reg_Adr);

**Пример:**ANL A, R3

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (A) & (Rn) , n=0,1,2,3,4,5,6,7  
**Машинный код:**

0 1 0 1 1 r r r

* [косвенно-регистровая](http://digteh.ru/MCS48/SysInstr.php#Indir_Reg_Adr);

**Пример:**ANL A, @R0

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (A) & ((Ri)) , i=0,1  
Машинный код:

0 1 0 1 0 0 0 i

* [непосредственная](http://digteh.ru/MCS48/SysInstr.php#Imm_Adr).

**Пример:**ANL A, #100

**Количество байт: 2  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 2  
(A) = (A) & (date 8)  
**Машинный код:**

0 1 0 1 0 0 1 1

i i i i i i i i

Для команды ANL возможна работа с портами и шиной данный в качестве приёмника информации. При этом источник информации использует [непосредственную](http://digteh.ru/MCS48/SysInstr.php#Imm_Adr) адресацию:

**Пример:**ANL BUS, #00010010B

**Количество байт: 2  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 2  
(BUS) = (BUS) & (direct)  
**Машинный код:**

1 0 0 1 1 0 0 0

d d d d d d d d

**Пример:**P1, #10000000b

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
(Pp) = (Pp) & (date 8), pp=1,2  
**Машинный код:**

1 0 0 1 1 0 p p

d d d d d d d d

**Примечание:**при выполнении команды ANL над содержимым внешнего порта используется значение из внутреннего регистра, а не данные, присутствующие на выводах микросхемы!

**ANLD P, A** - "логическое И" порта внешней микросхемы расширителя портов и аккумулятора **A.**

**Описание**: выполняет операцию побитового "логического И" над содержимым порта внешней микросхемы расширителя портов и аккумулятора. Результат  помещается в порт. Команда не воздействует на флаги. При этом как источник, так и приемник информации использует [неявную](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr) адресацию:

**Пример:**ANLD P6, A

**Количество байт: 1  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
(Pp) = (Pp) & (A), pp=4,5,6,7  
**Машинный код:**

1 0 0 1 1 1 p p

**CALL <addr 11>** - вызов подпрограммы.

**Описание**: вызывает подпрограмму, размещенную по указанному адресу <addr 11>. Команда увеличивает содержимое счетчика команд на 2 и затем помещает полученный результат в стек. Кроме программного счетчика в стеке сохраняется содержимое регистра флагов PSW. После это содержимое указателя стека SP увеличивается на 1.  Команда не воздействует на флаги.

**Пример:**CALL 311h

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
((SP)) = (PC), (PSW4-7)  
(SP) = (SP) + 1  
(PC10-0) = адрес подпрограммы (метки)  
**Машинный код:**

a10 a9 a8 1 0 1 0 0

a7 a6 a5 a4 a3 a2 a1 a0

**CLR** **A** - сброс аккумулятора

**Описание**: сбрасывает (записывает '0' во все биты аккумулятора). Команда не изменяет содержимое флагов. Команда использует [неявную](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr) адресацию:

**Пример:**CLR A      ;Записывает в аккумулятор число 00h

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = 0  
**Машинный код:**

0 0 1 0 0 1 1 1

**CLR C**- сбрасывает флаг переноса в нулевое значение.

**Описание**: сбрасывает (записывает '0') в флаг переноса. Команда использует [неявную](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr) адресацию:

**Пример:**CLR C ;Записывает во флаг переноса '0'

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(C) = 0

**CLR F0**- сбрасывает флаг пользователя F0 в нулевое значение.

**Описание**: сбрасывает (записывает '0') в флаг пользователя F0. Команда использует [неявную](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr) адресацию:

**Пример:**CLR F0 ;Записывает во флаг пользователя F0 '0'

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(F0) = 0  
**Машинный код:**

1 0 0 0 0 1 0 1

**CLR F1**- сбрасывает флаг пользователя F1 в нулевое значение.

**Описание**: сбрасывает (записывает '0') в флаг пользователя F1. Команда использует [неявную](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr) адресацию:

**Пример:**CLR F1 ;Записывает во флаг пользователя F0 '0'

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(F1) = 0  
**Машинный код:**

1 0 1 0 0 1 0 1

**CPL A** - каждый бит аккумулятора инвертируется.

**Описание**: каждый бит аккумулятора инвертируется, т.е. если в каком-либо из восьми бит аккумулятора записана '1', то в этот бит записывается '0' и наоборот.

**Пример:**CPL A

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = not(A)  
**Машинный код:**

0 0 1 1 0 1 1 1

**CPL C** - инвертируется флаг переноса 'C'.

**Описание**: флаг переноса 'C' инвертируется, т.е. если флаг переноса 'C' содержит '1', то в него записывается '0' и наоборот. Команда использует [неявную](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr) адресацию:

**Пример:**CPL C

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(C) = not(C)  
**Машинный код:**

1 0 1 0 0 1 1 1

**CPL F0** - инвертируется флаг пользователя F0 в нулевое значение.

**Описание:** флаг пользователя 'F0' инвертируется, т.е. если флаг пользователя 'F0' содержит '1', то в него записывается '0' и наоборот. Команда использует [неявную](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr) адресацию:

**Пример:** CLR F0 ;Инвертирует флаг пользователя F0

**Количество байт: 1  
Количество циклов: 1  
Действие команды:**  
(PC) = (PC) + 1  
(F0) = 0  
**Машинный код:**

1 0 0 0 0 1 0 1

**CPL F1** - инвертируется флаг пользователя F1 в нулевое значение.

**Описание:** флаг пользователя 'F1' инвертируется, т.е. если флаг пользователя 'F1' содержит '1', то в него записывается '0' и наоборот. Команда использует [неявную](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr) адресацию:

**Пример:** CLR F1 ;Инвертирует флаг пользователя F0

**Количество байт: 1  
Количество циклов: 1  
Действие команды:**  
(PC) = (PC) + 1  
(F1) = 0  
**Машинный код:**

1 0 1 0 0 1 0 1

**DA A**- десятичная коррекция аккумулятора при сложении двоично-десятичных чисел.

**Описание**: команда десятичной коррекции позволяет вести сложение чисел, представленных в упакованном двоично-десятичном коде. Команда применяется после инструкций **ADD** или **ADDC**. Инструкции (команды) **ADD** или **ADDC**позволяют суммировать сразу две тетрады, расположенные в одном байте.

Если биты аккумулятора с 0 по 3 содержат число большее 9 (xxxx1010:xxxx1111) или флаг вспомогательного переноса **AC** содержит '1', то к аккумулятору прибавляется число 6 для того, чтобы получить правильную двоично-десятичную цифру в младшей тетраде.

Если старшие биты аккумулятора (с 4 по 7) содержат число большее 9 (1010xxxx :1111xxxx) или флаг переноса **C** содержит '1', то к аккумулятору прибавляется число 60h для того, чтобы получить правильную двоично-десятичную цифру в старшей тетраде.

Если в результате операции **DA A** возникнет перенос, то флаг переноса **C** будет установлен в **1**. Единичный флаг переноса после операции **DA A** означает, что результат суммирования больше **100** и этим можно воспользоваться для суммирования многоразрядных двоично-десятичных чисел. Команда десятичной коррекции не воздействует на флаг переполнения **OV**.

**Примечание**: команда DA A не может преобразовать двоичное число из аккумулятора в двоично-десятичный вид или правильно скорректировать двоично-десятичное вычитание.

**Пример:** DA A

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
IF [[(A3-0) > 9] **or** [(AC) = 1]] THEN (A3-0) = (A3-0) + 6  
AND  
IF [[(A7-4) > 9] **or**[(C) = 1]] THEN (A7-4) = (A7-4) + 6  
**Машинный код:**

0 1 0 1 0 1 1 1

**DEC <байт>** - производит вычитание 1 из указанного операнда.

**Описание**: операнд уменьшается на 1. Если первоначально в ячейке было записано значение 00h, то в нее заносится значение 0FFh. Команда не воздействует на флаги.

Для команды DEC разрешены следующие режимы адресации:

* [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr).

**Пример:**DEC A

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (A) - 1  
**Машинный код:**

0 0 0 0 0 1 1 1

* [регистровая](http://digteh.ru/MCS48/SysInstr.php#Reg_Adr);

**Пример:**DEC R3

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(Rn) = (Rn) - 1, n=0,1,2,3,4,5,6,7  
**Машинный код:**

1

1 0 0 1 r r r

**DIS I** - запрет внешнего прерывания.

**Описание**: запрещает прерывание выполнения программы по сигналу низкого уровня на внешней ножке INT.

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
Запретить прерывания от внешней ножки INT  
**Машинный код:**

0 0 0 1 0 1 0 1

**DIS TCNTI** - запрет прерывания от таймера/счётчика.

**Описание**: запрещает прерывание выполнения программы по переполнению таймера/счётчика.

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
Запретить прерывания по переполнению таймера/счётчика  
**Машинный код:**

0 0 1 1 0 1 0 1

**DJNZ Rn,<адрес>**- вычитание 1 из указанной ячейки и переход, если результат не равен 0.

**Описание**: вычитание 1 из указанной во втором байте команды ячейки памяти и переход к вычисляемому по второму байту команды адресу, если результат вычитания не равен 0. Если первоначально в ячейке памяти содержится '0', то в результате выполнения операции туда будет записано число '0FFH'. Команда не воздействует на флаги. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Для команды DJNZ используется [регистровая](http://digteh.ru/MCS48/SysInstr.php#Reg_Adr) адресация:

**Пример:** LABEL1: DJNZ R3 ,LABEL1;

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
(Rn) = (Rn) - 1, n=0,1,2,3,4,5,6,7  
IF (Rn) < > 0, THEN (PC0..7) = Address  
**Машинный код:**

1 1 1 0 1 r r r

A d d r e s s

**EN I** - разрешение внешнего прерывания.

**Описание**: разрешает прерывание выполнения программы по сигналу низкого уровня на внешней ножке INT.

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
Разрешить прерывания от внешней ножки INT  
**Машинный код:**

0 0 0 0 0 1 0 1

**EN TCNTI** - разрешение прерываний от таймера/счётчика.

**Описание:**разрешает прерывание выполнения программы по переполнению таймера/счётчика.

**Количество байт: 1  
Количество циклов: 1  
Действие команды:**(PC) = (PC) + 1  
Разрешить прерывания от таймера/счётчика  
**Машинный код:**

0 0 1 0 0 1 0 1

**ENT0 CLK** - разрешение выдачи синхросигнала на вывод T0.

**Описание:**разрешает выдачу синхросигнала на вывод микросхемы T0.

**Количество байт: 1  
Количество циклов: 1  
Действие команды:**(PC) = (PC) + 1  
Разрешить  выдачу синхросигнала на вывод T0  
**Машинный код:**

0 1 1 1 0 1 0 1

**IN Pp**- пересылка данных из порта в аккумулятор.

**Описание:**копирует данные на выводах микросхемы в аккумулятор. Команда не воздействует на флаги и использует [неявную](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr) адресацию.

**Количество байт: 1  
Количество циклов: 1  
Действие команды:**(PC) = (PC) + 1  
(A) = (Pp), p=1,2  
**Машинный код:**

0 0 0 0 1 0 p p

**INC <байт>** - производит прибавление 1 к указанному операнду.

**Описание**: ячейка памяти адрес, которой указан во втором байте команды увеличивается на 1. Если первоначально в ячейке было записано значение 0FFh, то в нее заносится значение 00h. Команда не воздействует на флаги.

Для команды INC разрешены следующие режимы адресации байта - источника:

* [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr).

**Пример:**INC A

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (A) + 1  
**Машинный код:**

0 0 0 1 0 1 1 1

* [регистровая](http://digteh.ru/MCS48/SysInstr.php#Reg_Adr);

**Пример:**INC R3

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1(Rn) = (Rn) + 1, n=0,1,2,3,4,5,6,7  
**Машинный код:**

0 0 0 1 1 r r r

* [косвенно-регистровая](http://digteh.ru/MCS48/SysInstr.php#Indir_Reg_Adr);

**Пример:**INC @R0

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
((Ri)) = ((Ri)) + 1, i=0,1  
**Машинный код:**

0 0 0 1 0 0 0 i

**JB0 <адрес>** - переход если нулевой бит аккумулятора установлен.

**Описание**: Если нулевой бит аккумулятора равен единице, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:**JB0 LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
IF (ACC.0) = 1 THEN (PC0..7) = Address  
**Машинный код:**

0 0 0 1 0 0 1 0

A d d r e s s

**JB1 <адрес>**- переход если первый бит аккумулятора установлен.

**Описание:** Если первый бит аккумулятора равен единице, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:**JB1 LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды:**(PC) = (PC) + 2  
IF (ACC.1) = 1 THEN (PC0..7) = Address  
**Машинный код:**

0 0 1 1 0 0 1 0

A d d r e s s

**JB2 <адрес>** - переход если второй бит аккумулятора установлен.

**Описание:**Если второй бит аккумулятора равен единице, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:** JB2 LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды:**(PC) = (PC) + 2  
IF (ACC.2) = 1 THEN (PC0..7) = Address  
**Машинный код:**

0 1 0 1 0 0 1 0

A d d r e s s

**JB3 <адрес>** - переход если третий бит аккумулятора установлен.

**Описание:**Если третий бит аккумулятора равен единице, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:** JB3 LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды:**(PC) = (PC) + 2  
IF (ACC.3) = 1 THEN (PC0..7) = Address  
**Машинный код:**

0 1 1 1 0 0 1 0

A d d r e s s

**JB4 <адрес>** - переход если четвёртый бит аккумулятора установлен.

**Описание:**Если четвёртый бит аккумулятора равен единице, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:**JB4 LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды:**(PC) = (PC) + 2  
IF (ACC.4) = 1 THEN (PC0..7) = Address  
**Машинный код:**

1 0 0 1 0 0 1 0

A d d r e s s

**JB5 <адрес>**- переход если пятый бит аккумулятора установлен.

**Описание:**Если пятый бит аккумулятора равен единице, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:** JB5 LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды:**  
(PC) = (PC) + 2  
IF (ACC.5) = 1 THEN (PC0..7) = Address  
**Машинный код:**

1 0 1 1 0 0 1 0

A d d r e s s

**JB6 <адрес>** - переход если шестой бит аккумулятора установлен.

**Описание:**Если шестой бит аккумулятора равен единице, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:** JB6 LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды:**  
(PC) = (PC) + 2  
IF (ACC.6) = 1 THEN (PC0..7) = Address  
**Машинный код:**

1 1 0 1 0 0 1 0

A d d r e s s

**JB7 <адрес>** - переход если седьмой бит аккумулятора установлен.

**Описание:**Если седьмой бит аккумулятора равен единице, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:** JB7 LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды:**(PC) = (PC) + 2  
IF (ACC.7) = 1 THEN (PC0..7) = Address  
**Машинный код:**

1 1 1 1 0 0 1 0

A d d r e s s

**JC <адрес>** - переход, если бит переноса установлен.

**Описание**: Если бит переноса равен единице, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:**JС LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
IF (С) = 1 THEN (PC0..7) = Address  
**Машинный код:**

1 1 1 1 0 1 1 0

A d d r e s s

**JF0 <адрес>**- переход, если бит переноса установлен.

**Описание:**Если флаг пользователя F0 равен единице, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:** JF0 LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды:**(PC) = (PC) + 2  
IF (F0) = 1 THEN (PC0..7) = Address  
**Машинный код:**

1 0 1 1 0 1 1 0

A d d r e s s

**JF1 <адрес>**- переход, если бит переноса установлен.

**Описание:**Если флаг пользователя F1 равен единице, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:** JF0 LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды:**(PC) = (PC) + 2  
IF (F1) = 1 THEN (PC0..7) = Address  
**Машинный код:**

0 1 1 1 0 1 1 0

A d d r e s s

**JMP**<addr 11> - переход на метку.

**Описание:**осуществляет переход на метку, размещенную по указанному адресу <addr 11>. Команда не воздействует на флаги.

**Пример:**JMP 311h

**Количество байт: 2  
Количество циклов: 2  
Действие команды:**(PC10-0) = адрес метки  
**Машинный код:**

a10 a9 a8 0 0 1 0 0

a7 a6 a5 a4 a3 a2 a1 a0

**JMPP @A** - косвенный переход в текущей странице.

**Описание:**осуществляет переход на адрес, содержащийся в ячейке памяти программ на которую указывает содержимое аккумулятора. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с содержимым аккумулятора. Таким образом осуществляется переход в пределах страницы размером в 256 байт. Команда не воздействует на флаги и не изменяет содержимое аккумулятора.

**Пример:**JMPP @A

**Количество байт: 1  
Количество циклов: 2  
Действие команды:**(PC7-0) = ((A))  
**Машинный код:**

a10 a9 a8 0 0 1 0 0

a7 a6 a5 a4 a3 a2 a1 a0

**JNC <адрес>** - переход, если бит переноса сброшен.

**Описание**: Если бит переноса равен нулю, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:** JNС     LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
IF (С) = 0 THEN (PC0..7) = Address  
**Машинный код:**

1 1 1 0 0 1 1 0

A d d r e s s

# Описание машинных команд микроконтроллеров семейства MCS-48 (продолжение)

**JNI <адрес>** - переход, если сигнал на ножке запроса прерывания равен нулю.

**Описание**: Если на ножке запроса прерывания сигнал равен нулю, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:** JNI     LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
IF (INT) = 0 THEN (PC0..7) = Address  
**Машинный код:**

1 0 0 0 0 1 1 0

A d d r e s s

**JNT0 <адрес>** - переход, если сигнал ножке микроконтроллера T0 равен нулю.

**Описание**: Если на ножке микроконтроллера T0 сигнал равен нулю, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:** JNT0     LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
IF (T0) = 0 THEN (PC0..7) = Address  
**Машинный код:**

0 0 1 0 0 1 1 0

A d d r e s s

**JNT1 <адрес>** - переход, если бит переноса сброшен.

**Описание**: Если на ножке микроконтроллера T1 сигнал равен нулю, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги.

**Пример:** JNT1     LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
IF (T1) = 0 THEN (PC0..7) = Address  
**Машинный код:**

0 1 0 0 0 1 1 0

A d d r e s s

**JNZ <адрес>** - переход, если содержимое аккумулятора не равно нулю.

**Описание**: Если любой бит аккумулятора равен единице, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое аккумулятора.

**Пример:**JNZ     LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
IF (A) <> 0 THEN (PC0..7) = Address  
**Машинный код:**

1 0 0 1 0 1 1 0

A d d r e s s

**JTF <адрес>** - переход, если содержимое аккумулятора не равно нулю.

**Описание**: Если флаг переполнения таймера установлен, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги.

**Пример:** JTF LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
IF (A) <> 0 THEN (PC0..7) = Address  
**Машинный код:**

0 0 0 1 0 1 1 0

A d d r e s s

**JT0 <адрес>** - переход, если бит переноса сброшен.

**Описание**: Если на ножке микроконтроллера T0 сигнал равен единице, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:** JT0 LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
IF (T0) = 1 THEN (PC0..7) = Address  
**Машинный код:**

0 0 1 1 0 1 1 0

A d d r e s s

**JT1 <адрес>** - переход, если бит переноса сброшен.

**Описание**: Если на ножке микроконтроллера T1 сигнал равен единице, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое проверяемого бита.

**Пример:** JT1 LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
IF (T1) = 1 THEN (PC0..7) = Address  
**Машинный код:**

0 1 0 1 0 1 1 0

A d d r e s s

**JZ <адрес>** - переход, если содержимое аккумулятора не равно нулю.

**Описание**: Если  все биты аккумулятора равны нулю, то производится переход к указанному во втором байте команды адресу. В противном случае выполняется следующая команда. Новое значение программного счетчика при переходе вычисляется соединением содержимого старших бит программного счетчика с вторым операндом (вторым байтом команды). Таким образом осуществляется переход в пределах страницы размером в 256 байт. Если команда окажется на границе двух страниц, то переход осуществляется в пределах страницы, где расположен второй байт команды. Команда не воздействует на флаги и не изменяет содержимое аккумулятора.

**Пример:**JZ     LABEL1

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
IF (A) = 0 THEN (PC0..7) = Address  
**Машинный код:**

1 1 0 0 0 1 1 0

A d d r e s s

**MOV <приемник>,<источник>** - скопировать байтовую переменную.

**Описание**: переменная, указанная во втором операнде, копируется в ячейку, указываемую первым операндом. Байт источник не изменяется. Другие регистры и флаги не изменяются. Для каждого операнда используется свой вид адресации.

Команда допускает следующие виды адресации:

первый операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr), второй операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr) адресация;

**Пример:** MOV A, PSW

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (PSW)  
**Машинный код:**

1 1 0 0 0 1 1 1

**Пример:** MOV A, T

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (T)  
**Машинный код:**

0 1 0 0 0 0 1 0

**Пример:** MOV PSW, A

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(PSW) = (A)  
**Машинный код:**

1 1 0 1 0 1 1 1

**Пример:** MOV T, A

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(T) = (A)  
**Машинный код:**

0 1 1 0 0 0 1 0

первый операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr), второй операнд - [регистровая](http://digteh.ru/MCS48/SysInstr.php#Reg_Adr) адресация;

**Пример:** MOV A, R3

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (Rn) , n=0,1,2,3,4,5,6,7  
**Машинный код:**

1 1 1 1 1 r r r

первый операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr), второй операнд - [косвенно-регистровая](http://digteh.ru/MCS48/SysInstr.php#Indir_Reg_Adr) адресация;

**Пример:** MOV A, @R0

**Количество байт: 1  
Количество циклов: 1  
Действие команды:**(PC) = (PC) + 1  
(A) = ((Ri)) , i=0,1  
**Машинный код:**

1 1 1 1 0 0 0 i

первый операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr), второй операнд - [непосредственная](http://digteh.ru/MCS48/SysInstr.php#Imm_Adr) адресация;

**Пример:** MOV A, #38h

**Количество байт: 2  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 2  
(A) = (data 8)  
**Машинный код:**

0 0 1 0 0 0 1 1

i i i i i i i i

первый операнд - [регистровая](http://digteh.ru/MCS48/SysInstr.php#Reg_Adr), второй операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr) адресация;

**Пример:** MOV R3, A

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(Rn) = (A) , n=0,1,2,3,4,5,6,7  
**Машинный код:**

1 0 1 0 1 r r r

первый операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr), второй операнд - [непосредственная](http://digteh.ru/MCS48/SysInstr.php#Imm_Adr) адресация;

**Пример:** MOV R3, #38h

**Количество байт: 2  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 2  
(Rn) = (data 8) , n=0,1,2,3,4,5,6,7  
**Машинный код:**

1 0 1 1 1 r r r

i i i i i i i i

первый операнд - [косвенно-регистровая](http://digteh.ru/MCS48/SysInstr.php#Indir_Reg_Adr), второй операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr) адресация;

**Пример:** MOV @R0, A

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
((Ri)) = (A), i=0,1  
**Машинный код:**

1 0 1 0 0 0 0 i

первый операнд - [косвенно-регистровая](http://digteh.ru/MCS48/SysInstr.php#Indir_Reg_Adr), второй операнд - [непосредственная](http://digteh.ru/MCS48/SysInstr.php#Imm_Adr) адресация

**Пример:** MOV @R0, #80

**Количество байт: 2  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 2  
((Rn)) = (data 8) , n=0,1,2,3,4,5,6,7  
**Машинный код:**

1 0 1 1 0 0 0 i

i i i i i i i i

**MOVD A,<источник>** - считать порт микросхемы расширителя портов.

**Описание**: содержимое порта расширителя портов копируется в младшую тетраду аккумулятора. Старшая тетрада обнуляется. Команда не воздействует на флаги. В обоих операндах используется [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr) адресация.

**Пример:** MOVD A, P4

**Количество байт: 2  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 2  
(A) = (Pp), p=4,5,6,7  
**Машинный код:**

0 0 0 0 1 1 p p

**MOVP A, @A** - переслать байт из текущей страницы памяти программ в аккумулятор.

**Описание**: загружает аккумулятор константой из памяти программ. Адрес считываемого байта задаётся исходным содержимым аккумулятора. Команда не воздействует на флаги.

**Пример:** MOVP A, @A

**Количество байт: 1  
Количество циклов: 2  
Действие команды**:  
(PC0..PC7) = (A)  
(A) = ((PC))  
**Машинный код:**

1 0 1 0 0 0 1 1

**MOVP3 A, @A** - переслать байт из третьей страницы памяти программ в аккумулятор.

**Описание**: загружает аккумулятор константой из памяти программ. Адрес считываемого байта задаётся исходным содержимым аккумулятора. Команда не воздействует на флаги.

**Пример:** MOVP3 A, @A

**Количество байт: 1  
Количество циклов: 2  
Действие команды**:  
(PC0..PC7) = (A)  
(PC8..PC11) = 0011  
(A) = ((PC))  
**Машинный код:**

1 1 1 0 0 0 1 1

**MOVX <приемник>,<источник>** - переслать байтовую переменную во внешнюю память (из внешней памяти)

**Описание**: пересылает данные между аккумулятором и байтом внешней памяти.

В первом случае регистр R0 или R1 текущего банка регистров обеспечивает 8- битный адрес, который мультиплексируется с данными на выводах порта P0.

первый операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr), второй операнд - [косвенно-регистровая](http://digteh.ru/MCS48/SysInstr.php#Indir_Reg_Adr) адресация;

**Пример:** MOVX A, @R1 ;Скопировать данные из внешней памяти данных в аккумулятор

**Количество байт: 1  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 1  
(A) = ((Ri)), i=0,1  
**Машинный код:**

1 0 0 0 0 0 0 i

первый операнд - [косвенно-регистровая](http://digteh.ru/MCS48/SysInstr.php#Indir_Reg_Adr), второй операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr) адресация;

**Пример:** MOVX @R0, A ;Скопировать данные из аккумулятора во внешнюю память

**Количество байт: 1  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 1  
((Ri)) = (A), i=0,1  
**Машинный код:**

1 0 0 1 0 0 0 i

**NOP** - нет операции

**Описание**: Кроме программного счетчика не изменяет ни одного регистра, на флаги не воздействует.

**Пример:** NOP

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
**Машинный код:**

0 0 0 0 0 0 0 0

**ORL <приемник>,<источник>** - "логическое ИЛИ" для байтовых переменных

**Описание**: выполняет операцию побитового "логического ИЛИ" между указанными переменными. Результат сохраняется в приемнике. Команда не воздействует на флаги.

Для команды ORL с аккумулятором в качестве байта назначения разрешены следующие режимы адресации байта - источника:

первый операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr), второй операнд - [регистровая](http://digteh.ru/MCS48/SysInstr.php#Reg_Adr) адресация;

**Пример:**ORL A, R3

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (A) V (Rn) , n=0,1,2,3,4,5,6,7  
**Машинный код:**

0 1 0 0 1 r r r

первый операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr), второй операнд - [косвенно-регистровая](http://digteh.ru/MCS48/SysInstr.php#Indir_Reg_Adr) адресация;

**Пример:**ORL A, @R0

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (A) V ((Ri)), i=0,1  
**Машинный код:**

0 1 0 0 0 0 0 i

первый операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr), второй операнд - [непосредственная](http://digteh.ru/MCS48/SysInstr.php#Imm_Adr) адресация;

**Пример:**ORL A, #100

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
(A) = (A) V (date 8)  
**Машинный код:**

0 1 0 0 0 0 1 1

i i i i i i i i

**Пример:**ORL BUS, #100

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
(BUS) = (BUS) V (date 8)  
**Машинный код:**

1 0 0 1 1 0 0 0

i i i i i i i i

**Пример:**ORL P1, #10010000b

**Количество байт: 2  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 2  
(Pp) = (Pp) V (date 8), p=1,2  
**Машинный код:**

1 0 0 1 1 0 p p

i i i i i i i i

**Примечание:**при выполнении команды ORL над содержимым внешнего порта используется значение из внутреннего регистра, а не данные, присутствующие на выводах микросхемы!

**ORLD Pp, A** - логическое ИЛИ между портом расширителя и аккумулятором.

**Описание**: выполняет операцию побитового "логического ИЛИ" между портом микросхемы расширителя портов и аккумулятором. Результат сохраняется в порту. Команда не воздействует на флаги.

**Пример:**ORLD P5, A

**Количество байт: 1  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 1  
(Pp) = (Pp) V (A0...A3)  
**Машинный код:**

1 0 0 0 1 1 p p

**OUTL BUS, A** - стробируемый вывод данных из аккумулятора в порт BUS.

**Описание**: осуществляет стробируемый вывод данных из аккумулятора в порт BUS. Команда не воздействует на флаги.

**Пример:**OUTL BUS, A

**Количество байт: 1  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 1  
(BUS) = (A)  
**Машинный код:**

0 0 0 0 0 0 1 0

**OUTL Pp, A** - стробируемый вывод данных из аккумулятора в порт.

**Описание**: осуществляет вывод данных через порт. Команда не воздействует на флаги.

**Пример:**OUTL Pp, A

**Количество байт: 1  
Количество циклов: 2  
Действие команды**:  
(PC) = (PC) + 1  
(Pp) = (A), p=1,2  
**Машинный код:**

0 0 1 1 1 0 p p

**RET**- возврат из подпрограммы.

**Описание**: последовательно выгружает старший и младший байты счетчика команд из стека, уменьшая содержимое указателя стека на 1. Выполнение программы продолжается с нового, только что загруженного в счетчик команд адреса. Команда не воздействует на флаги.

**Пример:**RET

**Количество байт: 1  
Количество циклов: 2  
Действие команды**:  
(SP) = (SP) - 1  
(PC) = ((SP))  
**Машинный код:**

1 0 0 0 0 0 1 1

**RETR**- возврат из подпрограммы обслуживания прерывания и восстановление слова состояния программы.

**Описание**: последовательно выгружает старший и младший байты счетчика команд из стека, уменьшая содержимое указателя стека на 1. Кроме того, команда восстанавливает состояние логики прерываний, разрешая обработку следующего прерывания. Выполнение программы продолжается с того же адреса, что был в программном счетчике (PC) до начала обработки прерывания (следующий за командой, во время выполнения которой был обнаружен запрос на прерывание). Команда не воздействует на флаги. Если к этому моменту обнаруживается новый запрос на прерывание, то до нового вызова подпрограммы обслуживания прерывания выполняется одна команда из основной программы.

**Пример:**RETR

**Количество байт: 1  
Количество циклов: 2  
Действие команды**:  
(SP) = (SP) - 1  
(PC) = ((SP))  
(PSW4...PSW7) = ((SP))  
**Машинный код:**

1 0 0 1 0 0 1 1

**RL A**- сдвиг содержимого аккумулятора влево.

**Описание**: сдвигает восемь бит аккумулятора на один бит влево, бит 7 засылается на место бита 0. Команда не воздействует на флаги.

**Пример:**RL A

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(An+1) = (An) n=0..6  
(A0) = (A7)  
**Машинный код:**

1 1 1 0 0 1 1 1

**RLC A** - сдвиг содержимого аккумулятора влево через флаг переноса.

**Описание**: сдвигает восемь бит аккумулятора и флаг переноса на один бит влево. Содержимое флага переноса помещается на место бита 0 аккумулятора, а содержимое бита 7 аккумулятора переписывается во флаг переноса. На остальные флаги команда не воздействует.

**Пример:**RLC A

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(An+1) = (An) n=0..6  
(A0) = (C)  
(C) = (A7)  
**Машинный код:**

1 1 1 1 0 1 1 1

**RR A**- сдвиг содержимого аккумулятора вправо.

**Описание**: сдвигает восемь бит аккумулятора на один бит вправо, бит 0 засылается на место бита 7. Команда не воздействует на флаги.

**Пример:**RR A

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(An) = (An+1) n=0..6  
(A7) = (A0)  
**Машинный код:**

0 1 1 1 0 1 1 1

**RRC A** - сдвиг содержимого аккумулятора через флаг переноса.

**Описание**: сдвигает восемь бит аккумулятора и флаг переноса на один бит вправо. Содержимое флага переноса помещается на место бита 7 аккумулятора, а содержимое бита 0 аккумулятора переписывается во флаг переноса. На остальные флаги команда не воздействует.

**Пример:**RRC A

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(An) = (An+1) n=0..6  
(A7) = (C)  
(C) = (A0)  
**Машинный код:**

0 1 1 0 0 1 1 1

**SEL MB0** - выбор нулевого банка памяти программ.

**Описание**: сбрасывает старший бит счетчика команд в нулевое состояние. Команда используется для выбора нулевого банка памяти программ.

**Пример:** SEL MB0

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(PC11) = 0  
**Машинный код:**

1 1 1 0 0 1 0 1

**SEL MB1** - выбор первого банка памяти программ.

**Описание**: устанавливает старший бит счетчика команд в 1. Команда используется для выбора первого банка памяти программ.

**Пример:** SEL MB0

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(PC11) = 1  
**Машинный код:**

1 1 1 1 0 1 0 1

**SEL RB0** - выбор нулевого банка памяти программ.

**Описание**: сбрасывает флаг выбора банков в нулевое состояние. Команда используется для выбора нулевого банка регистров.

**Пример:** SEL RB0

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(BS) = 0  
**Машинный код:**

1 1 0 0 0 1 0 1

**SEL RB1** - выбор нулевого банка памяти программ.

**Описание**: устанавливает флаг выбора банков в 1. Команда используется для выбора первого банка регистров.

**Пример:** SEL RB0

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(BS) = 1  
**Машинный код:**

1 1 0 1 0 1 0 1

**STOP TCNT** - остановить таймер-счетчик.

**Описание**: прекращает подачу импульсов на вход таймера-счетчика.

**Пример:** STOP TCNT

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
остановить таймер-счетчик  
**Машинный код:**

0 1 1 0 0 1 0 1

**STRT CNT** - запустить счетчик.

**Описание**: разрешает подачу импульсов со входа микроконтроллера T1 на вход таймера-счетчика.

**Пример:** STRT CNT

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
запустить счетчик  
**Машинный код:**

0 1 0 0 0 1 0 1

**STRT T** - запустить таймер.

**Описание**: разрешает подачу импульсов с выхода предделителя на вход таймера-счетчика.

**Пример:** STRT CNT

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
запустить таймер  
**Машинный код:**

0 1 0 1 0 1 0 1

**SWAP A** - обмен тетрадами внутри аккумулятора.

**Описание**: команда осуществляет обмен между младшей (биты 0..3) и старшей (биты 4..7) тетрадами аккумулятора. (Тетрада - это четырехбитная переменная). Команду можно интерпретировать как циклический сдвиг на четыре бита. Команда не воздействует на флаги.

**Пример:**SWAP A

**Количество байт: 1  
Количество циклов: 1  
Действие команды:**(PC) = (PC) + 1  
(A3..A0) <=> (A7..A4) **Машинный код:**

0 1 0 0 0 1 1 1

**XCH A,<байт>** - обмен содержимого аккумулятора с переменной байтом.

**Описание**: команда загружает аккумулятор содержимым указанной переменной, в то же самое время первоначальное содержимое аккумулятора заносится по указанному адресу. Команда не воздействует на флаги.

В команде допускается следующие виды адресации:

первый операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr), второй операнд - [регистровая](http://digteh.ru/MCS48/SysInstr.php#Reg_Adr) адресация;

**Пример:** XCH A, R7

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) <=> (Rn) , n=0,1,2,3,4,5,6,7  
**Машинный код:**

0 0 1 0 1 r r r

первый операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr), второй операнд - [косвенно-регистровая](http://digteh.ru/MCS48/SysInstr.php#Indir_Reg_Adr) адресация;

**Пример:** XCH A, @R0

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC)= (PC) + 1  
(A) <=> ((Ri)) , i=0,1  
**Машинный код:**

0 0 1 0 0 0 0 i

**XCHD A, @Ri** - обмен цифры.

**Описание**: команда осуществляет обмен между младшей (биты 0..3) тетрадой (тетрада это четырехбитная переменная) аккумулятора, где обычно хранится двоично-десятичная цифра с тетрадой ячейки внутреннего ОЗУ. Используется косвенно регистровая адресация. Старшие тетрады (биты 4..7) обоих операндов не изменяются. Команда не воздействует на флаги.

**Пример:** XCHD A, @R0

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A3..A0) <=> ((Ri3..Ri0)) , i=0,1  
**Машинный код:**

0 0 1 1 0 0 0 i

**XRL <байт приемник>,<байт источник>** - исключающее ИЛИ для переменных байтов.

**Описание**: выполняет операцию побитового "исключающее логического ИЛИ" между указанными переменными. Результат сохраняется в приемнике. Команда не воздействует на флаги.

Для команды XRL с аккумулятором в качестве байта назначения разрешены следующие режимы адресации:

первый операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr), второй операнд - [регистровая](http://digteh.ru/MCS48/SysInstr.php#Reg_Adr) адресация;

**Пример:**XRL A, R3

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (A) **xor** (Rn) , n=0,1,2,3,4,5,6,7  
**Машинный код:**

1 1 0 1 1 r r r

первый операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr), второй операнд - [косвенно-регистровая](http://digteh.ru/MCS48/SysInstr.php#Indir_Reg_Adr) адресация;

**Пример**: XRL A, @R0

**Количество байт: 1  
Количество циклов: 1  
Действие команды**:  
(PC) = (PC) + 1  
(A) = (A) **xor** ((Ri)) , i=0,1  
**Машинный код:**

1 1 0 1 0 0 0 i

первый операнд - [неявная](http://digteh.ru/MCS48/SysInstr.php#Inher_Adr), второй операнд - [непосредственная](http://digteh.ru/MCS48/SysInstr.php#Imm_Adr) адресация;

**Пример:**XRL A, #100

**Количество байт: 2  
Количество циклов: 1  
Действие команды**:  
(PC)= (PC) + 2  
(A) = (A) **xor** (date 8)  
**Машинный код:**

1 1 0 1 0 0 1 1  
i i i i i i i i