## Тема 11. Команди логічних операцій та зсувів.

### *Команды логических операций*

В системах команд всех ЭВМ есть команды логических операций. Они выполняют поразрядные операции инвертирования, И, ИЛИ, исключительно ИЛИ (сложения по *mod 2* ), а также циклический сдвиг аккумулятора. При этом инвертирование заменяет каждый бит его дополнением; операция И дает в результате бит, равный единице, если соответствующие биты обоих операторов содержат единицу; операция ИЛИ - бит, равный единице, если, соответствующие биты в одном или обоих операнда содержат единицы; операция исключающего ИЛИ - бит, равный единице, если только соответствующий бит одного операнда содержит единицу.

Большинство применений логических операций связано с изменением, обнулением /сбросом/ и проверкой бит в байте. Эти действия выполняются с помощью логических операций над нужным байтом и вторым байтом - маской. Такая операция называется маскированием.

Логические команды сведены в табл. 11.1 и содержат команды И, ИЛИ, ИЛИ ИСКЛЮЧАЮЩЕЕ,НЕ (инверсия) и сдвига. Здесь именно аккумулятор составляет ядро

большинства операций. Как и при арифметических командах, способ адресации и здесь влияет на способ и место нахождения других данных в системе.

Таблица11.1. Команды логических операций типовогоМП

| Операция | Адресация | Мнемоника | КОП | Байты | Формат команды | Символика | Установка индикаторов |
|---|---|---|---|---|---|---|---|
| А И данные | Непосредственная | AN1 | E6 | 2 | коп данные | (A) ← (A) • (байт 2) | Z CY — Сброс |
| А И L | Регистровая | ANA L | A5 | 1 | коп | (A) ← (A) • (L) | Z CY — Сброс |
| А И Н | | ANA H | A4 | 1 | коп | (A) ← (A) • (H) | Z CY — Сброс |
| А И LOC (HL) | Косвенная регистровая | ANA M | A6 | 1 | коп | (A) ← (A) • (H) | Z CY — Сброс |
| А ИЛИ данные | Непосредственная | OR1 | F6 | 2 | коп данные | (A) ← (A) + (байт 2) | Z CY — Сброс |
| А ИЛИ L | Регистровая | ORA | B5 | 1 | коп | (A) ← (A) + (L) | Z CY — Сброс |
| А ИЛИ Н | | ORA H | B4 | 1 | коп | (A) ← (A) + (H) | Z CY — Сброс |
| А ИЛИ LOC (HL) | Косвенная регистрация | ORAM | B6 | 1 | коп | (A) ← (A) + ((H)(L)) | Z CY — Сброс |
| А ИЛИ ИСКЛЮЧАЮЩЕЕ данные | Непосредственная | XRI | EE | 2 | коп данные | (A) ← (A) ○ (байт 2) | Z CY — Сброс |
| А ИЛИ ИСКЛЮЧАЮ-ЩЕЕ А | Регистровая | XRA A | AF | 1 | коп | (A) ← (A) ○ (A) Сброс А | Z = 1 CY — Сброс |
| А ИЛИ ИСКЛЮЧАЮ-ЩЕЕ L | » | XRAL | AD | 1 | коп | (A) ← (A) ○ (L) | Z CY — Сброс |
| А ИЛИ ИСКЛЮЧАЮ ЩЕЕ Н | » | XRA H | AC | 1 | коп | (A) ← (A) ○ (H) | Z CY — Сброс |
| А ИЛИ ИСКЛЮЧАЮ-ЩЕЕ LOC (HL) | Косвенная регистровая | XRAM | AE | 1 | коп | (A) ← (A) + ((H)(L)) | Z CY — Сброс |
| ИнвертироватьА | Неявная | CMA | 2F | 1 | коп | (A) ← (A') | |
| Сдвиг А вправо | » | RAR | 1F | 1 | коп | ... →(CY) →(A) → (CY)→ ... | CY |
| Сдвиг А влево | » | RAL | 17 | 1 | коп | ... ← (CY) ← (A) ← (CY) ← ... | CY |

*1. Логическое «И», логическое умножение или конъюнкция.*

Конъюнкция - это сложное логическое выражение, которое считается истинным в том и только том случае, когда оба простых выражения являются истинными, во всех остальных случаях данное сложеное выражение ложно. Логическое «И» обычно обозначается следующими символами: &, *, или же отсутствием символа между двумя логическими высказываниями. Высказывание A&B (AB, AxB) истинно тогда и только тогда, когда истины оба высказывания A и B. Это обстоятельство удобно выразить в виде таблицы истинности для конъюнкции

Таблица истинности для конъюнкции

| **A** | **B** | **F** |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

Рассмотрим выполнение типовым микропроцессором команды И непосредственно, рис.11.1,а.

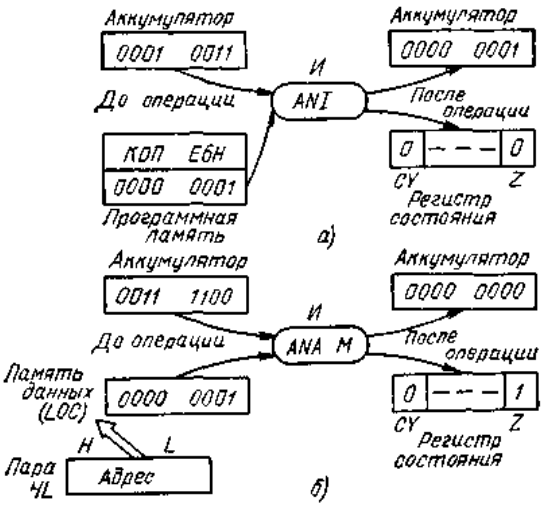



Рис. 11.1. Команда И с непосредственной адресацией (а) и команда ANA M *(б)*

Содержимое аккумулятора (0001 0011) подвержено операции И побитно. Согласно таблице истинности для операции И только самые младшие биты каждого числа равны 1, следовательно, результатом будет 0000 0001, он помещается в аккумулятор. Согласно последней колонке табл. 11.1 результатом всех операций И будет сброс индикатора переноса, мы это видим также на рис. 11.1, а. Результат операции *И* проверяется с целью определения — не нуль ли он, и если нет, то индикатор нуля сбрасывается в 0. Отметим использование точки (•) в табл. 11.1 в колонке символов для обозначения операции И.

На рис. 11.1,*б* приведен другой пример

использования команды И, в этом случае — И косвенной адресации (мнемоника ANA *M).* Содержимое аккумулятора подвержено операции И (бит с битом) с содержимым ячейки памяти, указанной парой *HL.*После операции 0011 1100 и0000 0001 полученный результат 0000 0000 помещен в аккумулятор. Индикатор переноса сбрасывается (СБРОС) согласно табл. 11.1. Помещенный в аккумулятор результат проверяется, и, поскольку он равен 0, индикатор нуля устанавливается в 1.

Внимательно рассмотрим рис. 11.1,б. В этих двух примерах второй операнд — 0000 0001, он используется как *маска.* Маска 0000 0001 на рис. 11.1, а и *б* может быть использована для сброса в 0 семи старших бит или, с учетом наличия индикатора нуля, для тестирования значений 0 или 1 в позиции младшего бита аккумулятора (в этом случае на единственную 1 надевается маска 0000 0001). Но нужно быть осторожным. Заметим, что содержимое аккумулятора изменилось после операции И. Некоторые микропроцессоры снабжены специальными командами тестирования битов, которые выполняют операции И с содержимым аккумулятора и с маской байта без изменения содержимого всего аккумулятора, изменяя состояние индикаторов.

***2.****Логическое «ИЛИ», логическое сложение или дизъюнкция:*

Логическое «ИЛИ», называемое также дизъюнкцией и логическим сложением. - это сложное логическое выражение, которое истинно, если хотя бы одно из простых логических выражений истинно и ложно тогда и только тогда, когда оба простых логических выраженныя ложны. Обычно данная операция обозначается символом ∨ (A∨B) Операция дизъюнкции может быть представлена следующей таблицей истинности:

Таблица истинности для дизъюнкции

| **A** | **B** | **F** |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

Четыре команды ИЛИ (см. табл. 11.1) выполняются с содержимым аккумулятора и содержимым какой-либо другой ячейки памяти и регистра. На рис. 11.2 приведен пример операции ИЛИ, когда содержимым аккумулятора будет 1100 1100, тогда как регистр *L*содержит 0000 1111. Результат—

1100 1111. Числа подвержены операции ИЛИ побитно согласно таблице истинности ИЛИ Содержимое 0000 1111 регистра *L*может рассматриваться как маска, которая всегда будет обращать 4 младших бита в 1111. Отметим использование логического знака (+) для обозначения операции ИЛИ в колонке символов в табл. 11.1.

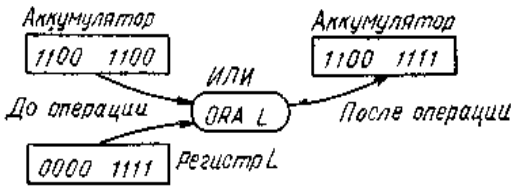



Рис. 11.2. Команда (A) ИЛИ (L)

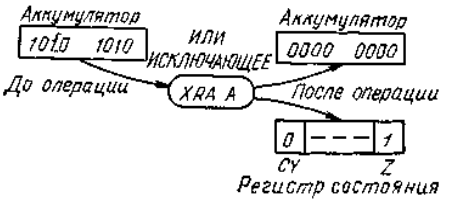
***3.*** *Логическая команда «исключающегоИЛИ»*

В табл. 10.1 приведены пять команд «исключающего ИЛИ» для типового микропроцессора. Эти команды производят поразрядное двоичное сложение операндов. Операция «исключающего ИЛИ», обозначаемую как $\oplus$. Так запись $A \oplus B$ означает либо A либо B, но не оба вместе. Таблица истинности для «исключающего ИЛИ» имеет вид:

| A | B | F |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

Команда XRA применяется для инвертирования определённых битов слова с помощью слова маски. Например, для инвертирования седьмого и первого битов маска имеет вид 10000010. Тогда не зависимо от того какое число будет в этих разрядах после поразрядного сложения, согласно таблицы истинности, будет инвертирование в указанных разрядах.

На рис. 11.1 показан процесс выполнения команды «исключающего ИЛИ» *A с A.* Результатом «исключающего ИЛИ» 1010 1010 с самим собой (рис. 11.3) будет 0000 0000. Выполнение этой операции любого числа с самим собой всегда дает результат 0000 0000, индикатор нуля всегда устанавливается в 1, что означает нулевое содержимое индикатора, а в соответствии с табл. 11.1 индикатор переноса *CY* всегда будет сброшен в 0.




Рнс. 11.3. Команда XRA *A*

***4.*** *Логическое отрицание «НЕ» или инверсия.*

Инверсия - это сложное логическое выражение. Если исходное логическое выражение истинно, то результат отрицания будет ложным, и наоборот, если исходное логическое выражение ложно, то результат отрицания будет истинным.

Таблица истинности для инверсии

| **A** | **не A** |
|---|---|
| 1 | 0 |
| 0 | 1 |

Операция инвертирования осуществляется с содержимым аккумулятора посредством мнемокоманд: CMA, CMP, CMP M.

Логические команды позволяют установить в единицу, сбросить в ноль, инвертировать и проверить интересующие нас биты. ***Установка бита*** в единицу производится формированием *маски*, которая определяет позицию требуемого бита. Маска для

установки бита – это байт с нулями во всех битах, кроме искомого. Затем выполняется операция ИЛИ с содержимым аккумулятора и полученной маской. Аналогично, ***сброс*** бита в ноль осуществляется операцией И над маской, с единицами во всех битах, кроме требуемого, обозначенного нулем. После этих операций следует команда перехода по условию состояния флага Z.

Для примера запишем программы изменения содержимого пятого бита регистра B:

```
;Установка в "1"     ;Сброс в "0"        ;Инверсия бита
MVI A,00100000B     MVI A,11011111B     MVI A,00100000B
ORA B               ANA B               XRA B

MOV B,A             MOV B,A             MOV B,A
```

### *Команды сдвига.*

Восьмиразрядный микропроцессор имеет четыре команды циклического сдвига. Операндом их является содержимое аккумулятора, в котором формируется результат. Сдвиги выполняются влево и вправо на один разряд. В зависимости от того, что помещается в освобождающийся при сдвиге бит и как используется выдвигающийся бит и «флаг переноса» вводятся несколько команд сдвигов.

RLC – циклический сдвиг содержимого аккумулятора на одну позицию влево. Младший бит и флаг C принимают значение вытесненного бита, т.е. бывшего старшего бита;

RRC – циклический сдвиг содержимого аккумулятора на одну позицию вправо. Старший бит и флаг C принимают значение вытесненного бита, т.е. бывшего младшего бита;

RAL – циклический сдвиг содержимого аккумулятора на одну позицию влево вместе с флагом C. В младшем бите устанавливается содержимое флага C;

RAR – циклический сдвиг содержимого аккумулятора на одну позицию вправо вместе с флагом C. В старшем бите устанавливается содержимое флага C.

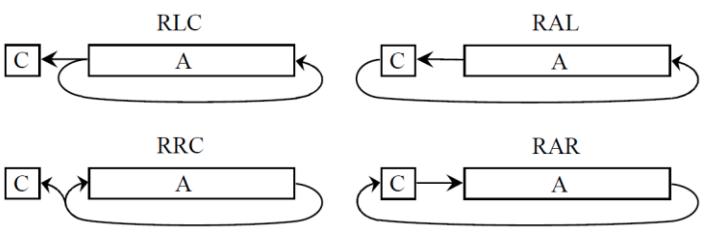



Рис.11.4.Выполнение команд циклического сдвига

Рассмотрим приведенный на рис. 11.5 пример использования команды циклического сдвига вправо с переносом: содержимое

аккумулятора (0011 0001) сдвинуто на одну позицию вправо и его младший бит (1 в этом примере) передается в позицию бита индикатора переноса, тогда как имевшийся там бит занимает позицию старшего бита аккумулятора, в котором содержится 0001 1000 после завершения операции. Индикатор переноса установится в 1, индикатор нуля не изменится.

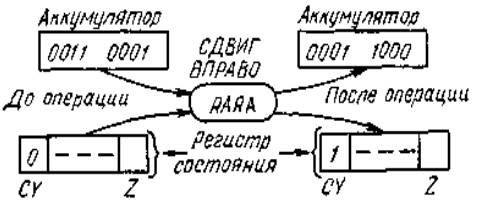



Рис.11.5. Команда RAR *A*

Используя одну или несколько команд циклического сдвига, можно тестировать весь заданный состав бит, а индикатор переноса может быть сброшен или установлен. Индикатор переноса может быть тестирован затем командой условного ветвления. Тест паритета является другим приложением использования команд сдвига. Паритет двоичного числа определяется числом содержащихся в нем единиц: четный паритет — общее число единиц четное; нечетный паритет — общее число единиц нечетное.

Команды сдвига оперируют только данными

аккумулятора и не требуют других операндов, расположенных в памяти или регистрах. Таким образом, команды логических операций используются для манипуляции с переменными по законам алгебры логики. Они могут быть использованы для тестирования и сравнения бит.

### *Примеры на использование логических команд сдвига*

*Пример 1.* Из содержимого ячейки *40H* выделить младшие 4 бита (младшую шестнадцатеричную цифру ) и переслать их по адресу *41H*. Старшие 4 бита (старшую шестнадцатеричную цифру) содержимого ячейки *40H* переслать в младшие разряды ячейки *42H*.

Программа:

| Адрес | Ассемблерный код | Комментарий |
|---|---|---|
| 00 | *LXI H, 40H* | ; Загрузить |
| 01 | | непосредственный |
| 02 | | ; адрес в регистровую пару |
| *03* | | ; *HL* |
| 04 | *MOV A, M* | ; Загрузить в *A* данные по адресу *HL* |
| 05 | *MOV B, A* | ; Запомнить данные в регистре *B* |
| 06 | *ANI* 00001111B | ; Маскировка старшей цифры ; |
| 07 | *INX H* | ; Увеличить на 1 |

| | | |
|---|---|---|
| | | содержимое *HL* . . |
| 08 | *MOV M, A* | ; Запомнить младшую цифру |
| 09 | *MOV A, B* | ; Переслать исходное число из *B* в *A* |
| 0A | *RRS* | ; Четырехкратный |
| 0B | *RRS* | ; циклический |
| 0C | *RRS* | ; сдвиг |
| 0D | *RRS* | ; вправо |
| 0E | *ANI 00001111B* | ; Маскировка младшей цифры |
| 0F | | ; |
| 10 | *INX H* | ; Увеличить на 1 содержимое *HL* |
| 11 | *MOV M, A* | ; Запомнить старшую цифру |
| 12 | *END* | ; Конец. |

*Пример 2.* В ячейке 0040 находится слово, биты 2 и 3 которого необходимо селективно установить, бит 6 сбросить, а бит 5 - изменить, а затем проверить. В ячейках 0030-0032 хранятся маски, соответствующие кодам *OC, BF* и 20.

Ячейки 0030, 0031, 0032, 0040 соответственно имеют следующие метки*:* *MASKS, MASKS+1, MASKS+2* и *CTRL* .

Программа:

| Адрес | Ассемблерный код | Комментарий |
|---|---|---|
| 00 | *LXI H, MASKS* | ; Загрузить непосредственный |
| 01 | | ;адрес маски в регистровую |

```
02                          ; пару HL
03      LDA   (CTRL)        ; Загрузить в A
                              содержимое
04                          ; ячейки CTRL
05                          ;
06      ORA   M             ; Установить бит 2 и
                              3
07      INX   H             ; Инкремент до
        MASKS+1
08      ANA   M             ; Сбросить бит 6
09      INX   H             ; Инкремент до
        MASKS+2
0A      XRA  M              ; Изменить бит 5
0B      STA   CTRL          ; Запомнить в CTRL
0C                          ;
0D                          ;
0E      ANA   M             ; Проверить бит 5
0F        *                 ; Конец.
```

Таким образом, с помощью команд логических операций и сдвигов могут быть выполнены логические операции И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ (сложение по модулю 2) с использованием двух операндов. Один из операндов всегда размещается в аккумуляторе. Результат ***выполнения команды*** фиксируется в аккумуляторе.

Команда ANA

Команда ANA осуществляет логическую операцию И над содержимым аккумулятора и содержимым одного из регистров РОН, аккумулятора и ячейки М

ANA r; A & r→ A

ANA M; A & M→ A

***Команда ORA***

Команда ORA реализует логическую ИЛИ над содержимым аккумулятора и содержимым одного из регистров РОН, аккумулятора и ячейки М

ORA r; A Ú r→A

ORA M A Ú M→A

***Команда XRA***

Команда XRA осуществляет логическую операцию ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым аккумулятора и содержимым одного из регистров РОН, аккумулятора и ячейки М

XRA r; A Å r→ A

XRA M; A Å M→ A

***Команды ANI, ORI, XRI***

Эти команды отличаются от команд ANA, ORA, XRA тем, что второй операнд следует непосредственно за кодом операции.

ANI data8; A & data8→ A

ORI data8; A Ú data8→ A

XRI data8; A Å data8→ A

***Команды CMP, CPI***

Команда CMP используется для сравнения двух чисел, одно из которых находится в аккумуляторе, а другое в одном из регистров РОН, аккумуляторе или ячейке М. При сравнении одно из чисел вычитается из другого числа. В соответствии с результатом

формируются признаки регистра признаков. Содержимое аккумулятора при этом не изменяется. CMP r; A – r

CMP M; A – M

CPI data8 A – data8

Разница между командами CMP и SUB в том, что при выполнении команды CMP результат операции не фиксируется в аккумуляторе.

Внимание! Команды логических операций и сравнения модифицируют регистр признаков.

***Команда CMA***

Команда CMA используется для инвертирования содержимого аккумулятора.

CMA; A → A

***Внимание!*** Команда CMA не модифицирует регистр признаков.

***Команды STC, CMC***

STC; 1 → CY

CMC; CY → CY

Команда STC устанавливает признак переноса CY. Команда CMC инвертирует признак переноса.

Команды сдвига RLC, RRC, RAL, RAR

Если надо произвести операции сдвига над данными, то их необходимо предварительно поместить в аккумулятор. Операндом однобайтных команд сдвига является содержимое аккумулятора, в котором формируется результат. Сдвиги выполняются

влево (RLC, RAL) и вправо (RRC, RAR) только на один разряд. Выполнение команд сдвига поясняется на рисунке ниже.

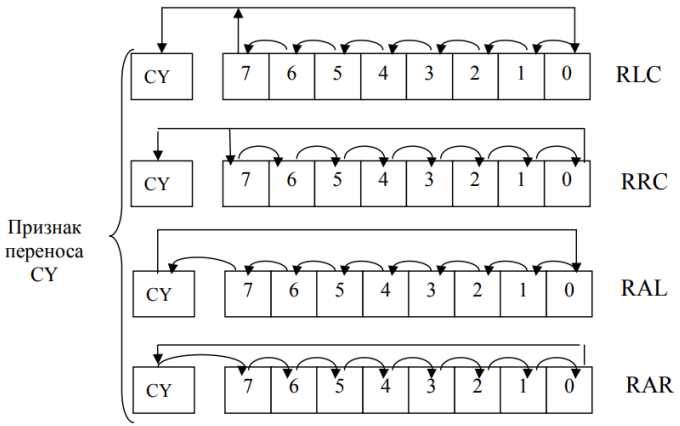



Рис. Команды сдвига

***Состав команд прочих операций***

Эти команды составляют последнюю категорию, которыми наделен типовой микропроцессор. Они сведены в табл. 6.8 и содержат команды помещения в стек, извлечения из стека, отсутствия операции и команду остановки. При их выполнении индикаторы не изменяются.

Команды помещения в стек и извлечения из него уже упоминались в § 5.5. Они используются всегда парно, так как то, что в стек помещается, должно быть из него извлечено. Они широко распространены при использовании подпрограмм. Команда поместить в стек содержимое аккумулятора*A* и индикаторов могла бы быть, например, первой командой подпрограммы, приведенной на рис. 6.32. Она сохранила бы содержимое аккумулятора и индикаторов независимо от подпрограммы. Точно перед операцией возврата на рис.

6.32 команда извлечь из стека*А* и индикаторы восстановила бы начальное содержимое аккумулятора и индикаторов.

Таблица 6.8. *Прочие команды типового микропроцессора*

| Операция | Адресация | Мнемоника | КОП | Байт | Т команд | Симнолика |
|---|---|---|---|---|---|---|
| Поместить в стекАи индикаторы | Косвенная регистровая | PUSH PSW | F5 | 1 | коп | ((SP) - 1) ← (A)<br>((SP) - 2) ← (индикаторы)<br>(SP) ← (SP) - 2 |
| Поместить в стек HL | То же | PUSH H | E5 | 1 | коп | ((SP) - 1) ← (H)<br>((SP) - 2) ← (L)<br>(SP) ← (SP) - 2 |
| Извлечь из стека А и индикатора | » | POP PSW | Fl | 1 | коп | (индикаторы) ← ((SP))<br>(A) ← ((SP) + 1)<br>(SP) ← (SP) + 2 |
| Извлечь из стекла HL | » | POP H | El | 1 | коп | (L) ← ((SP))<br>(H) ← ((SP) + 1)<br>(SP) ← (SP) + 2 |
| Нет операций | Неявная | NOP | 00 | 1 | коп | (L) ← ((SP))<br>(H) ← ((SP) + 1) |
| Останов | Неотделим | HLT | 76 | 1 | коп | (SP) ← (SP) + 2<br>(PC) ← (PC) +1 |

Рассмотрим первую команду поместить в стек*А* и индикаторы (PUSH *PSW).*Часть *PSW*соответствует слову состояния программы которое в данном случае является содержимым аккумулятора и регистра состояния (индикаторов). Команда PUSH *PSW*является однобайтовой, содержимое аккумулятора помещается первым, а регистра состояния —

вторым. Для более подробного ознакомления с командами помещения и извлечения из стека (PUSH) (POP) соответственно следует обратиться к § 5.5.

Команда НЕТ ОПЕРАЦИЙ соответствует отсутствию всякого выполнения операций в течение 1 или 2 мкс. Это однобайтовая команда, единственным эффектом которой является инкремент счетчика команд. Никакой другой регистр не затрагивается. Эта команда используется как дополнение (когда одна или две команды отменены в ходе наладки) и связывает две части программы так, чтобы МП мог обратиться от одной к другой. Она может также служить для ввода интервала времени в цикл временной задержки.

Команда ОСТАНОВ используется в конце программы для остановки микропроцессора. В этом случае только СБРОС или команда вызова прерывания может позволить новый запуск типового микропроцессора.