**Макроассемблер**

**М80**

**(С) 1995,1996 Орионсофт**

Содержание

|  |  |  |
| --- | --- | --- |
|   |  Аннотация. | 4 |
| 1.  |  Назначение и условия применения макроассемблера. | 4 |
| 2.  |  Элементы и основные конструкции языка. | 4 |
| 2.1.  |  Основные элементы языка.  | 4 |
| 2.1.1.  |  Символы. | 4 |
| 2.1.2.  |  Числа, константы и строки. | 4 |
| 2.1.3.  |  Символические имена. | 5 |
| 2.2.  |  Выражения. | 5 |
| 2.2.1.  |  Арифметические и логические операции | 5 |
| 2.2.2.  |  Абсолютные и перемещаемые имена. | 6 |
| 2.2.3.  |  Внешние ссылки | 6 |
| 2.3.  |  Синтаксис ассемблерных предложений | 7 |
| 2.3.1.  |  Метка. | 7 |
| 2.3.2.  |  Оператор | 7 |
| 2.3.3.  |  Операнды и примечания. | 7 |
| 2.4.  |  Псевдокоманды. | 8 |
| 2.4.1.  |  Псевдокоманды перемещения. | 8 |
| 2.4.1.1.  |  ASEG | 8 |
| 2.4.1.2.  |  COMMON | 8 |
| 2.4.1.3.  |  CSEG | 9 |
| 2.4.1.4.  |  DSEG | 9 |
| 2.4.2.  |  DB - определить байт | 9 |
| 2.4.3.  |  DC - определить символ | 9 |
| 2.4.4.  |  DS - определить область. | 10 |
| 2.4.5.  |  DW - определить слово. | 10 |
| 2.4.6.  |  END. | 10 |
| 2.4.7.  |  ENTRY/PUBLIC | 10 |
| 2.4.8.  |  EQU. | 11 |
| 2.4.9.  |  EXT/EXTRN. | 11 |
| 2.4.10.  |  INCLUDE. | 11 |
| 2.4.11.  |  NAME | 11 |
| 2.4.12.  |  ORG - задать счетчик адреса. | 12 |
| 2.4.13.  |  PAGE | 12 |
| 2.4.14.  |  SET. | 12 |
| 2.4.15.  |  TITLE. | 12 |
| 2.4.16.  |  SUBTTL | 12 |
| 2.4.17.  |  .COMMENT | 13 |
| 2.4.18.  |  .PRINTX. | 13 |
| 2.4.19.  |  .RADIX | 13 |
| 2.4.20.  |  .REQUEST | 13 |
| 2.4.21  |  Условные псевдокоманды.  | 13 |
| 2.4.21.1  |  .ELSE. | 14 |
| 2.4.23.  |  Перемещение перед загрузкой | 15 |
| 2.5.  |  Макрокоманды и блоки. | 15 |
| 2.5.1.  |  Терминология. | 16 |
| 2.5.2.  |  Блок REPT-ENDM. | 16 |
| 2.5.3.  |  Блок IRP-ENDM. | 16 |
| 2.5.4.  |  Блок IRPC-ENDM. | 17 |
| 2.5.5.  |  MACRO | 17 |
| 2.5.6.  |  EXITM | 18 |
| 2.5.7.  |  LOCAL | 18 |
| 2.5.8.  |  Специальные макрооператоры и формы. | 19 |
| 3.  |  Руководство по эксплуатации компилятора M80 | 21 |
| 3.1.  |  Обращение к компилятору. Входные и выходные данные. | 21 |
| 3.2.  |  Ключи компиляции. | 21 |
| 3.3.  |  Формат листинга | 22 |
| 3.4.  |  Сообщения об ошибках. | 23 |
| 4.  |  Программа вывода перекрестных ссылок. | 24 |

**Аннотация**

Данное руководство содержит сведения о макроассемблере M80. Описаны элементы макроассемблера, псевдокоманды и макрокоманды. Приведено руководство по эксплуатации макроассемблера M80. В нем указано, как вызвать макроассемблер, как с помощью ключей управлять процессом компиляции. Приведен список сообщений об ошибках.

Перед изучением данного руководства, необходимо ознакомиться с системой команд микропроцессора КР580ВМ80А.

**1. Назначение и условия применения макроассемблера**

Макроассемблер - это компилятор с символического машинно-ориентированного языка (языка ассемблера). Он преобразует исходную программу, написанную на, написанную на языке макроассемблера, в перемещаемый модуль. В дальнейшем с помощью компоновщика L80 модуль может быть скомпонован (один или с другими модулями) в программу, готовую к выполнению.

Применение макросредств позволяет наиболее часто встречающиеся участки программы оформлять в виде макрокоманд.

Использование макроассемблера значительно облегчает написание программ и позволяет наиболее эффективно использовать все ресурсы ЭВМ.

Макроассемблер находится на диске в виде файла с именем M80.COM

**2. Элементы и основные конструкции языка**

**2.1. Основные элементы языка**

**2.1.1. Символы**

Все сложные элементы языка М80 формируются из символов.

В качестве символов могут использоваться:

-буквы латинского алфавита от A до Z ;

-цифры от 0 до 9 ;

-специальные символы + - , : ; ( ) ' # $ пробел ;

**2.1.2. Числа, константы и строки**

Числа - это беззнаковые 16-битные величины. Число всегда вычисляется в системе счисления по умолчанию, если оно не задано в специальном виде:

 NNNNNNNN B - двоичное ( где N - 0 или 1 )

 NNNN O или NNNN Q - восьмеричное ( где N - 0...7 )

 NNNN H или NNNN X - шестнадцатеричное ( где N - 0...9,A,B,C,D,E,F ),если первый символ не цифра, то перед ним должен стоять 0 0NNNN H или 0NNNN X

По умолчанию все числа считаются в десятичной системе счисления. Этот режим можно изменить. С помощью команды .RADIX можно выбрать любое основание системы счисления от 2 до 16. Переполнение при вычислении числа игнорируется и значащими являются младшие 16 бит.

Символьная константа - это строка, содержащая один или два символа в апострофах.

Значением символьной константы, содержащей один символ, является величина численного значения соответствующего кода, принятого в операционной системе (ОС) CP/M. Например, значением константы "A" будет 41H. Значением символьной константы, содержащей два символа, будет двухбайтовое число, причем в старшем байте будет значение первого символа , а в младшем байте - значение второго символа. Например, значением константы "AB" будет число 41H\*256+42H.

Строка - это набор символов, заключенных в кавычки или апострофы. Кавычки также могут быть элементом строки, но при этом они должны удваиваться. Например, предложение

 DB "Я""ИДУ""ГУЛЯТЬ"

запомнит строку

 Я"ИДУ"ГУЛЯТЬ

Если между кавычками нет символов, то такая строка будет пустой строкой.

**2.1.3. Символьные имена**

Символические имена могут содержать любое количество символов, но значащими являются первые 6 символов. В именах допустимы следующие символы:

 буквы от A до Z , цифры от 0 до 9 , $ , . , ? , @

Символическое имя не может начинаться с цифры.

При обработке имен буквы из нижнего регистра переводятся в буквы верхнего регистра.

**2.2. Выражения**

**2.2.1. Арифметические и логические операции**

В выражениях допустимы следующие операции, приведенные в порядке убывания приоритета их выполнения.

 NUL

 LOW,HIGH

 \*,/,MOD,SHR,SHL

 Унарный минус

 +,-

 EQ,NE,LT,LE,GT,GE

 NOT

 AND

 OR,XOR

Для изменения порядка выполнения операций надо использовать скобки. При вычислении выражений сначала выполняются операции, имеющие более высокий приоритет. Все подвыражения, содержащие операции с более высоким приоритетом, вычисляются в первую очередь.

Все операции , кроме +,-,\*,/, должны быть отделены от операндов по крайней мере одним пробелом. Операции выделения байта (HIGH, LOW) выделяют старшие или младшие 8 бит из абсолютной 6-битной величины. Если в качестве операнда используется перемещаемая величина, то HIGH и LOW интерпретируются как операции, работающие с адресом ноль.

**2.2.2. Абсолютные и перемещаемые имена**

Все символические имена, используемые в качестве операндов в выражениях, есть либо абсолютные имена, либо перемещаемые имена. Перемещаемые имена подразделяются на программные перемещаемые имена, перемещаемые имена данных и COMMON (п. 2.4.1.).

Все символические имена, ассемблированные для псевдокоманд ASEG, CSEG (по умолчанию) или DSEG, относятся к абсолютным, перемещаемым программным или перемещаемым данным соответственно.

Количество имен COMMON в программе зависит от числа блоков COMMON, поименованных в программе с помощью псевдокоманды COMMON. Два символических имени COMMON считаются разными, если они присутствуют в разных блоках COMMON.

При любой операции, кроме сложения и вычитания, оба операнда должны быть абсолютными.

Для сложения применяются следующие правила:

1) по крайней мере один из операндов должен быть абсолютным;

2) абсолютное + перемещаемое = перемещаемое

Для вычитания действуют правила:

1) перемещаемое - абсолютное = перемещаемое

2) перемещаемое - перемещаемое = абсолютное

Причем в 2) оба имени должны быть перемещаемыми именами одного типа.

Для каждого промежуточного шага при вычислении выражения должны выполняться вышеуказанные правила, иначе будет выдано сообщение об ошибке. Например, если FOO,BAZ и ZAZ - три символических перемещаемых имени, то выражение:

 FOO + BAZ - ZAZ

выработает сообщение об ошибке, т.к. на первом шаге будут складываться два перемещаемых имени. Этого можно избежать, используя скобки:

 FOO + (BAZ - ZAZ)

**2.2.3. Внешние ссылки**

Символическое имя может быть внешним (EXTRN) или нет (см.п. 2.2.9.). Внешнее имя ассемблируется в 2 байта. При использовании внешних имен в выражениях должны выполняться

следующие правила:

1) EXTRN допустимы только при сложении и вычитании;

2) если в выражении используется имя типа EXTRN, то результат всегда будет EXTRN;

3) при сложении любой операнд (но не оба одновременно) может быть EXTRN;

4) при вычитании только первый операнд может быть EXTRN.

**2.3. Синтаксис ассемблерных предложений**

Ассемблерная программа состоит из последовательности предложений. Каждое предложение имеет следующую структуру:

 метка: (оператор) (операнды) ;примечание

Совсем необязательно, чтобы предложение начиналось с метки.

**2.3.1. Метка**

Метка - это символическое имя, после которого следует двоеточие. Если метка присутствует, то она должна быть первым элементом предложения. Значение метки есть текущее значение счетчика адреса. Если после метки стоят два двоеточия, то такая метка объявляется внутренней ( PUBLIC ) и к ней можно обращаться из другой программы.

**2.3.2. Оператор**

Следующим элементом после метки или первым, если нет метки идет оператор. В качестве оператора может быть:

 1) мнемокод микропроцессора КР580ВМ80А ;

 2) макровызов ;

 3) псевдокоманда ;

 4) выражение.

Если в поле оператора стоит выражение, то ассемблер рассматривает его как псевдокоманду DB.

**2.3.3. Операнды и примечания**

 Следующим элементом предложения являются операнды. В зависимости от оператора в поле операндов может быть один, два или несколько операндов.

 В качестве операндов можно использовать мнемокоды команд микропроцессора КР580ВМ80А. При этом следует учитывать, что допустимым операндом будет только первый байт команды.

 Пример.

 MVI A,JMP

 ADI CPI

 MVI B,RNZ

 CPI INX H

 ACI LXI B

 MVI C,MOV A,B

Если в операнд будет включено более одного байта (CPI 5, LXI B,LAM,JMP A5), то будет выдано сообщение об ошибке.

Примечания являются последним элементом предложения. Примечания всегда начинаются с символа ; и заканчиваются CR (ВК).

**2.4. Псевдокоманды**

**2.4.1. Псевдокоманды перемещения**

Отличительной особенностью макроассемблера М80 является создание им перемещаемых модулей. Перемещаемость модуля облегчает его отладку и тестирование, позволяет компоновать программу из нескольких модулей, написанных на разных языках. Кроме того, программу можно разбивать на сегменты и указать, какие сегменты должны размещаться только в ОЗУ (сегменты данных), а какие могут быть занесены в ПЗУ (сегменты содержащие программу). Эти сегменты создаются с помощью псевдокоманд DSEG и CSEG. Псевдокоманда ASEG используется при создании неперемещаемых (абсолютных) кодов. Псевдокоманда COMMON создает общие области данных для каждого поименованного в программе блока COMMON.

Для каждого сегмента программы ASEG, CSEG или DSEG заводится свой счетчик адреса, который приостанавливает отсчет при смене типа сегмента и возобновляет отсчет, когда снова встречается соответствующий сегмент. Значение счетчика адреса в любом сегменте можно изменить с помощью псевдокоманды ORG (п.2.4.12).

По умолчанию всегда действует псевдокоманда CSEG.

**2.4.1.1. ASEG**

 Формат псевдокоманды:

 ASEG

ASEG устанавливает значение счетчика адреса на абсолютный сегмент в памяти. Значение счетчика будет продолжением предыдущего ASEG (по умолчанию 0), если после ASEG не стоит ORG.

**2.4.1.2. COMMON**

 Формат псевдокоманды:

 COMMON /"имя блока"/

COMMON устанавливает счетчик адреса на адрес выбранного общего блока памяти. Для совместимости с оператором COMMON в ФОРТРАНЕ этот адрес всегда указывает на начало области. Если "имя блока" опущено или состоит из пробелов, то блок COMMON считается пустым.

**2.4.1.3. CSEG**

Формат псевдокоманды:

 CSEG

CSEG устанавливает значение счетчика адреса на смещение относительно начала сегмента памяти, содержащего коды команд. Значение адреса будет соответствовать значению предыдущего CSEG (по умолчанию 0), если только после CSEG не стоит псевдокоманда ORG. Для ассемблера CSEG является псевдокомандой по умолчанию.

**2.4.1.4. DSEG**

Формат псевдокоманды:

 DSEG

DSEG устанавливает значение счетчика сегмента адреса на относительный адрес сегмента памяти, содержащего данные. Значение адреса будет соответствовать адресу последнего DSEG (по умолчанию 0), если не было псевдокоманды ORG.

**2.4.2. DB - определить байт**

Формат псевдокоманды:

DB "выражение","выражение",... или

DB "строка" ["строка"...]

Аргументами для DB могут быть выражения или строки. DB запоминает значения выражений или символов в строках в последовательных адресах памяти, начиная с адреса, указанного в счетчике адреса. Значение выражения должно помещаться в одном байте. Если старший байт результата 0 или 255, то сообщение об ошибке не выдается. В противном случае выдается ошибка А.

В выражениях не могут присутствовать строки, содержащие более 2-х символов. Старший бит каждого байта, отведенного под соответствующий символ, равен нулю.

 Примеры:

 0000' 41 42 DB 'AB'

 0002' 42 DB 'AB' AND 0FFH

 0003' 41 42 43 DB 'ABC'

**2.4.3. DC - определить символ**

 Формат псевдокоманды:

 DC "строка"

DC запоминает символы, заданные в "строке", последовательно, начиная с адреса, указанного в счетчике адреса. В отличие от DB в байте, содержащем последний символ строки, в старшем разряде стоит 1. Если строка пустая, то выдается сообщение об ошибке.

**2.4.4. DS - определить область**

 Формат псевдокоманды:

 DS "выражение"

DS резервирует область памяти. Значение выражения показывает, сколько байтов должно быть зарезервировано. Все имена, присутствующие в выражении, должны быть определены к моменту вычисления выражения при первом проходе макроассемблера. В противном случае при первом проходе появится флаг V, а при втором проходе - флаг U. Если втором проходе флаг U не появится, то все равно в модуле может быть ошибка, т.к. DS обрабатывается при первом проходе.

**2.4.5. DW - определить слово**

 Формат псевдокоманды:

DW "выражение","выражение",...

DW последовательно записывает значения выражений в ячейки памяти, начиная с адреса, содержащегося в счетчике адреса. Значения выражений есть двухбайтовые величины.

**2.4.6. END**

 Формат псевдокоманды:

 END "выражение"

Оператор END указывает на конец программы. Если присутствует "выражение", то оно указывает на стартовый адрес программы. Если "выражение" отсутствует, то компоновщику L80 стартовый адрес не передается.

 \*) Примечание.

 Если ассемблерная программа является основной программой при компоновке, то в ней должен быть указан стартовый адрес. В противном случае компоновщик L80 выдаст сообщение об ошибке "нет стартового адреса". Если же ассемблерная программа является подпрограммой по отношению к программе, написанной, например, на ФОРТРАНЕ, то стартовый адрес можно не задавать, т.к. он будет задан в основной программе.

**2.4.7. ENTRY/PUBLIC**

 Формат псевдокоманды:

 ENTRY "имя","имя",... или

 PUBLIC "имя","имя",...

ENTRY или PUBLIC объявляют все имена, присутствующие в списке, как внутренние, доступные извне. После этого к таким именам можно обращаться из других программ. Все имена в списке ENTRY/PUBLIC должны быть определены, иначе будет сообщение об ошибке U. Имена, объявленные ENTRY или PUBLIC, заносятся в список глобальных имен и используются компоновщиком L80 при удовлетворении внешних ссылок.

**2.4.8. EQU**

 Формат псевдокоманды:

 "имя" EQU "выражение"

По команде EQU символическому имени присваивается значение выражения. Если "выражение" имеет EXTRN, то выдается сообщение об ошибке.

**2.4.9. EXT/EXTRN**

 Формат псевдокоманды:

 EXT "имя","имя",... или

 EXTRN "имя","имя",...

EXT или EXTRN объявляют указанные имена внешними (EXTRN), т.е. определенными в другой программе. Если имя, указанное в списке, определено в данной программе, то будет выдано сообщение об ошибке M. Если заканчивается двумя знаками числа (##), то оно также объявляется как внешнее.

**2.4.10. INCLUDE**

 Формат псевдокоманды:

 INCLUDE "имя файла"

Три псевдокоманды INCLUDE, $INCLUDE и MACLIB эквивалентны. Псевдокоманда INCLUDE позволяет включать в текст исходной программы, записанные в других файлах. "имя файла" – это имя файла, содержащего включаемую программу. Тип файла должен

быть MAC.

Файл, присутствующий в псевдокоманде INCLUDE, открывается и транслируется в текущую программу сразу же за псевдокомандой INCLUDE.

При выводе листинга в каждой строке, содержащей элемент включенного файла, сразу же после кода команды стоит знак "+".

Вложение псевдокоманд INCLUDE не допускается. Если файл, указанный в INCLUDE, не существует, то будет выдано сообщение об ошибке "V" и данная псевдокоманда игнорируется.

**2.4.11. NAME**

 Формат псевдокоманды:

 NAME ('имя программы')

NAME задает имя программы. Значащими в "имени" являются только первые шесть знаков. Имя можно задавать и с помощью псевдокоманды TITLE. Если в программе отсутствует NAME и TITLE, то в качестве имени берется имя исходного файла.

**2.4.12. ORG - задать счетчик адреса**

 Формат псевдокоманды:

 ORG "выражение"

Счетчику адреса присваивается выражение. Все имена в выражении должны быть определены при первом проходе.

**2.4.13. PAGE**

 Формат псевдокоманды:

 PAGE "выражение"

Псевдокоманда PAGE дает указание ассемблеру начать новую страницу вывода листинга. Если в псевдокоманде присутствует "выражение", то оно задает число строк на странице. Это число должно быть в диапазоне от 1 до 255. По умолчанию это число равно 50.

**2.4.14. SET**

Формат псевдокоманды:

"имя" SET "выражение"

SET осуществляет то же действие, что и EQU, но имя в псевдокоманде SET может быть определено ранее. Тогда после выполнения SET имя будет переопределено.

**2.4.15. TITLE**

 Формат псевдокоманды:

 TITLE "текст"

TITLE задает заголовок программы, который может печататься в первой строке каждой страницы. В тексте программы может быть только одна псевдокоманда TITLE иначе будет сообщение об ошибке Q. Если не задана псевдокоманда NAME , то первые шесть символов заголовка зададут имя модуля. Если в программе нет ни NAME, ни TITLE, то имя модуля будет задано из имени исходного файла.

**2.4.16. SUBTTL**

 Формат псевдокоманды:

 SUBTTL "текст"

Эта псевдокоманда выводит заголовок после заголовка, выведенного псевдокомандой TITLE. Длина "текста" не должна превышать 60 знаков. В программе может быть любое количество SUBTTL.

**2.4.17. .COMMENT**

 Формат псевдокоманды:

 .COMMENT "разделитель""текст""разделитель"

Первый символ, идущий после .COMMENT и отличный от пробела, считается разделителем. Текст после разделителя считается примечаниями и может занимать произвольное количество строк. Примечания заканчиваются, когда встречается второй разделитель - тот же самый символ, который был использован в качестве первого разделителя.

**2.4.18. .PRINTX**

Формат псевдокоманды:

.PRINTX "разделитель""текст""разделитель"

Первый символ после .PRINTX, отличный от пробела, считается разделителем. Текст, идущий после разделителя, во время трансляции выводится на терминал до тех пор, пока не встретится второй разделитель - тот же самый символ.

 \*) Примечание.

.PRINTX выводит листинг при каждом проходе ассемблера, поэтому, если листинг нужен только при одном каком либо проходе, то следует использовать псевдокоманды IF1 или IF2.

**2.4.19. .RADIX**

 Формат псевдокоманды:

 .RADIX "выражение"

По умолчанию система счисления всех констант - десятичная. Псевдокоманда .RADIX позволяет изменить основание системы счисления по умолчанию (от 2 до 16). "выражение" в .RADIX всегда в десятичной системе счисления независимо от основания системы счисления, действующей в данный момент.

**2.4.20. .REQUEST**

 Формат псевдокоманды:

 .REQUEST "имя файла","имя файла",...

Эта псевдокоманда передает запрос компоновщику L80 на поиск неудовлетворенных внешних имен в файлах, указанных в списке. Имена файлов в списке не должны содержать типов файла и ссылок на диски. L80 ведет поиск файла с заданным именем и с типом - REL на диске по умолчанию.

**2.4.21. Условные псевдокоманды**

Список условных псевдокоманд

|  |  |
| --- | --- |
| IF/IFT "выражение" | Истина, если "выражение" не равно 0 |
| IFE/IFF "выражение" | Истина, если "выражение" равно 0 |
| IF1 | Истина при первом проходе |
| IF2 | Истина при втором проходе |
| IFDEP "символ" | Истина, если "символ" определен или объявлен EXTERNAL |
| IFDEF "символ" | Истина, если "символ" не определен и не объявлен EXTERNAL |
| IFB "аргумент" | Истина, если "аргумент" есть пробел. Аргумент должен быть в угловых скобках |
| IFNB "аргумент" | Истина, если "аргумент" не пробел. Используется для проверки некоторых параметров. Аргумент должен быть в угловых скобках. |
| IFIDN "аргумент1", | Истина, если строка "аргумент1" идентична строке "аргумент2". Аргументы должны быть в угловых скобках. |
| "аргумент2" |
| IFDIF "аргумент1", | Истина, если строка "аргумент1" отличается от строки "аргумент2". |
| "аргумент2" |

 Все условные псевдокоманды имеют формат:

 IF XXX [аргумент]

 .

 .

 .

 [ELSE

 .

 .

 ]

 ENDIF

Допускается вложение условных псевдокоманд, т.е. псевдокоманда внутри псевдокоманды. Во избежание ошибок все аргументы условных псевдокоманд должны быть известны при первом проходе.

Для IF, IFT, IFF и IFE выражения должны содержать только определенные символы и все выражения должны быть абсолютными.

**2.4.21.1. ELSE**

Каждая псевдокоманда IF должна иметь соответствующую ей псевдокоманду ENDIF, заканчивающую условие.

**2.4.22. Псевдокоманды, управляющие выводом листинга**

Выводом листинга в файл можно управлять с помощью двух псевдокоманд:

 .LIST и .XLIST

Если листинг не выводится, то данные псевдокоманды игнорируются.

По умолчанию действует псевдокоманда .LIST, инициирующая вывод листинга. Если встречается псевдокоманда .XLIST, то вывод листинга прекращается, пока не встретится псевдокоманда .LIST.

Вывод условных блоков, имеющих значение ложь, употребляется тремя псевдокомандами:

.SFCOND подавление вывода условных блоков, имеющих значение "ложь";

.LFCOND восстановить вывод условных блоков, имеющих значение "ложь";

.TFCOND управлять выводом условных блоков, имеющих значение "ложь", в зависимости от ключа /X.

Вывод таблицы перекрестных управляется псевдокомандами .CREF и .XCREF. Если кросс-средства не используются, то эти псевдокоманды игнорируются. Условие по умолчанию - .CREF. Если встречается псевдокоманда .XCREF, то вывод перекрестных ссылок прекращается до тех пор, пока снова не встретится псевдокоманда .CREF.

Вывод расширений MACRO/REPT/IRP/IRPC управляется тремя псевдокомандами:

.LALL выводит полный макротекст для всех расширений;

.SALL выводит только сгенерированные объектные коды;

.XALL выводит сгенерированные объектные коды и соответствующий им текст (условие по умолчанию).

**2.4.23. Перемещение перед загрузкой**

С помощью двух псевдокоманд: .PHAZE и .DEPHAZE некоторые участки программ могут находится в одной области, а выполняться лишь в другой, заданной, области.

Псевдокоманда .PHAZE обозначает начало такого участка, а псевдокоманда .DEPHAZE - конец участка.

 Пример.

 0000' .PHAZE

 0100 CD 0106 F00: CALL BAZ

 0103 C3 0007 JMP Z00

 0106 C9 BAZ: RET

 0007' C3 0005 Z00: JMP 5

Все метки внутри блока .PHAZE получают абсолютные значения в зависимости от значения счетчика адреса .PHAZE. Коды, однако, загружаются в текущую область памяти (т.е. с 0000' в этом примере). Позже их можно переслать по адресу 100H и выполнить.

**2.5. Макрокоманды и блоки**

Макросредства ассемблера MACRO включают в себя три макрокоманды повторения (REPT, IRP, IRPC) и операцию макроопределения (MACRO). Каждая макрокоманда должна заканчиваться псевдокомандой ENDM.

**2.5.1. Терминология**

Для описания макрокоманд используются следующие термины:

1) "фор.пар." - формальный параметр. Все формальные параметры являются допустимыми символическими именами, которые появляются в теле макрорасширения;

2) "список форм.пар." - список формальных параметров, разделенных запятыми;

3) "список арг." - список аргументов, разделенных запятыми. Список аргументов всегда заключен в угловые скобки. Если в угловых скобках нет символа или между двумя запятыми нет символа, то это нулевой аргумент в списке. При вложении угловых скобок (скобки в скобках) каждый внутренний уровень скобок удаляется при использовании соответствующего аргумента. Строка в кавычках является допустимым аргументом;

4) "список пар." - список параметров, разделенных запятыми.

**2.5.2. Блок REPT-ENDM**

 Формат макрокоманды:

 REPT "выражение"

 .

 .

 ENDM

Вычисляется значение "выражения" и группа предложений, заключенная между REPT и ENDM, повторяется полученное число раз. "выражение" не может содержать внешние и неопределенные имена.

 Пример.

 X SET 0

 REPT 10 ;генерирует DB 1 - DB 10

 X SET X+1

 DB X

 ENDM

**2.5.3. Блок IRP-ENDM**

 Формат макрокоманды:

 IRP "фор.пар.",<список арг.>

 .

 .

 .

 ENDM

Число аргументов в списке определяет, сколько раз будет повторена группа предложений, заключенная между IRP и ENDM. При каждом повторении вместо всех вложений формального параметра подставляется очередное значение из списка аргументов. Если список аргументов пуст (<>), то группа предложений обрабатывается 1 раз и формальный параметр удаляется из всех вхождений.

 Пример.

 IRP X, <1,2,3,4,5,6,7,8,9,10>

 DB X

 ENDM

 Результат будет аналогичен предыдущему примеру.

**2.5.4. Блок IRPC-ENDM**

 Формат макрокоманды:

 IRPC "фор.пар.",строка (или "строка")

 .

 .

 .

 ENDM

Количество знаков в строке указывает, сколько раз будет повторена группа предложений между IRPC и ENDM. В данном случае угловые скобки не обязательны. При каждом повторении вместо формального параметра будет подставляться очередной знак из строки.

 Пример.

 IRPC X, 0123456789

 DB X+1

 ENDM

Результат аналогичен двум предыдущим примерам.

**2.5.5. MACRO**

 Формат макрокоманды:

 "имя" MACRO "список форм.пар."

 .

 .

 .

 ENDM

С помощью MACRO пользователь может создавать свои макрокоманды и помещать их в различные участки программы.

"имя" - это символическое имя, по которому впоследствии будут вызывать данную макрокоманду.

"список форм.пар." - это список параметров, которые будут изменяться (замещаться фактическими параметрами) при каждом вызове макрокоманды. Предложения между MACRO и ENDM составляют тело макрокоманды.

Форма обращения к макрокоманде следующая:

 "имя" "список пар."

где "имя" - это имя присвоенное MACRO в макроопределении;

"список пар." - это список фактических параметров, которые при вызове макрокоманды заменят соответствующие формальные параметры.

Если число фактических параметров превышает количество формальных параметров, то лишние фактические параметры игнорируются. Если же число параметров меньше числа формальных параметров, то вместо недостающих фактических параметров будут подставлены нули.

 Пример.

 F00 MACRO X

 Y SET 0

 REPT X

 Y SET Y+1

 DB Y

 ENDM

 ENDM

При обращении F00 10 эта макрокоманда сгенерирует те же предложения, что и в предыдущих примерах.

**2.5.6. EXITM**

Псевдокоманда EXITM используется для завершения REPT/IRP/IRPC или вызова MACRO. При выполнении EXITM макрорасширение немедленно выводится, а все оставшееся расширение или повторение не генерируется. Если блок, содержащий EXITM, входит в другой блок, то внешний уровень продолжает расширяться.

**2.5.7. LOCAL**

 Формат команды:

 LOCAL "список фор.пар."

Эта псевдокоманда допустима только внутри макроопределения MACRO. При выполнении LOCAL ассемблер создает уникальное символическое имя для каждого формального параметра из списка и подставляет потом это символическое имя при каждом вхождении данного формального параметра в макрорасширение. Обычно, эти уникальные символические имена используются для задания меток внутри MACRO. Это исключает возможность появления многократно определенных меток при неоднократных вызовах данной макрокоманды. Символические имена, создаваемые при помощи LOCAL, находятся в диапазоне от ..0001 до ..FFFF. Поэтому пользователю в своих программах следует избегать имен типа ..NNNN. Если в макрокоманде есть LOCAL , то LOCAL должна быть в первом приложении макроопределения.

**2.5.8. Специальные макрооператоры и формы**

& амперсанд используется для конкатенации текста или символических имен. Формальный параметр, представляющий собой строку в апострофах, не будет заменен в расширении, если перед ним не будет стоять амперсанд. Для образования символического имени из текста и формального параметра между ними надо поставить &;

 Пример.

 ERG MACRO X

 ERROR&X: PUSH BX

 MVI BX,'&X'

 JMP ERROR

 ENDM

В этом примере вызов ERG A создаст следующую последовательность предложений:

 ERROR&A: PUSH B

 MVI B,'A'

 JMP ERROR

\*) Примечание.

Два символических имени считаются одинаковыми, если одно имя отличается от другого только наличием амперсанда. Так символическое имя ERRORA эквивалентно имени ERROR&A.

;; Если в макрокоманде перед комментариями стоят две точки с запятой, то эти комментарии не появляются в макрорасширениях даже, если есть указание .LALL;

$ ссылка на текущее значение счетчика адреса;

! символ, стоящий после восклицательного знака вводится литерально;

NUL это оператор, возвращающий значение "истина", если его аргумент есть нуль. Остаток строки после NUL рассматривается как аргумент для NUL. Условие IF NUL аргумент будет "ложным", если при расширении первый символ аргумента будет отличен от точки с запятой или "CR" (ВК). Этот оператор рекомендуется использовать для проверки наличия нулевых параметров в условных псевдокомандах IFB и IFNB;

% этот оператор может использоваться только в аргументе макрокоманды. % преобразует следующее за ним выражение (обычно символическое имя) в число в текущей системе счисления. Во время макрорасширения полученное число подставляется в формальный параметр. Использование % позволяет вызывать макрокоманду с числом (обычно макровызов - это вызов с текстом из аргументов). Выражение, следующее за % должно удовлетворять тем же требованиям, что и в псевдокоманде DS.

 Пример.

Нормально LB, являющееся аргументом для MACLAB, должно было подставляться аргументом MACRO. % преобразует LB в число и вместо Y подставляется уже число. Без % результат трансляции был бы 'ERROR LB'.

 MACLAB MACRO Y

 ERR&Y: DB 'ERROR &Y',0

 ENDM

 MACERR MACRO X

 LB SET 0

 REPT X

 LB SET LB+1

 MACLAB %LB

 ENDM

 ENDM

После макровызова MACERR 3 получим:

 ERR1: DB 'ERROR 1',0

 ERR2: DB 'ERROR 2',0

 ERR3: DB 'ERROR 3',0

 TYPE - оператор TYPE возвращает байт, который описывает две характеристики аргумента: перемещаемость и является ли аргумент внешним (EXTRN) или нет. Аргументом для TYPE может быть любое выражение (строковое, числовое, логическое). Если выражение не верно, то TYPE вернет ноль. Структура возвращаемого байта следующая:

Младшие два бита определяют перемещаемость аргумента следующим образом:

 0 абсолютный;

 1 программный перемещаемый;

 2 данных перемещаемый;

 3 COMMON перемещаемый.

Старший бит (80H) - бит EXTRN. Когда выражение содержит внешний символ, то бит равен 1.

 Бит определенности - (20H). Когда выражение определено, то этот бит равен 1. Если же выражение не определено или содержит EXTRN, то этот бит равен 0.

 TYPE обычно используют внутри макрокоманд, когда необходимо проверить тип аргумента.

**3. Руководство по эксплуатации компилятора M80**

**3.1. Обращение к компилятору. Входные и выходные данные**

Исходными данными для компилятора является файл, содержащий программу написанную на макроассемблере, и имеющий тип MAC. В результате работы компилятор создает файл типа REL, содержащий модуль в перемещаемой форме. Кроме того, по желанию пользователя может быть создан файл с листингом и файл перекрестных ссылок.

Загрузка и запуск макроассемблера осуществляется одной из следующих команд:

 1) M80

 2) M80 "командная строка"

В режиме (1) M80 выводит на экран промпт "\*" и ждет ввода командной строки. В режиме (2) командная строка вводится сразу после имени M80.

 Формат командной строки:

 "имя 1","имя 2"="имя 3"

 где "имя 1" - имя файла, в который будет записан перемещаемый модуль;

 "имя 2" - устройство, на которое будет выводится листинг. Таким устройством может быть консоль (TTY:), логическое устройство (LST:) или на файл на диске;

 "имя 3" - имя файла, содержащего исходный текст программы.

Если "имя 1" и "имя 2" опущены, то по умолчанию берется имя файла исходного текста. Тип файла в "имя 1" всегда REL, в "имя 2", если задан файл всегда PRN, а в "имя 3" всегда MAC.

Если перемещаемый модуль и листинг не нужны, то слева от знака равенства надо поставить запятую.

 Примеры:

|  |  |
| --- | --- |
| \*=SOURSE.MAC  | Транслировать программу SOURSE.MAC и поместить объектный код в файл SOURSE.REL |
| \*,LST:=TEST  | Транслировать программу TEST.MAC и вывести листинг на устройство LST; |
| \*SMALL,TTY:=TEST  | Транслировать программу TEST.MAC, объектный файл записать в SMALL.REL, а листинг вывести на консоль. |

**3.2. Ключи компиляции**

В командной строке может быть задано несколько ключей, которые будут влиять на формат выводимого листинга.

|  |  |
| --- | --- |
| Ключ | Действие |
|  /O  |  Печатать в листинге все адреса и т.п. в восьмеричной системе счисления |
|   |
|  /H  |  Печатать в листинге все адреса и т.п. в шестнадцатеричной системе счисления (действует по умолчанию) |
|   |
|  /X  | Подавить вывод листинга условий, значение которых есть "ложь" |
|  /P  | Создать объектный файл |
|  /L  | Создать файл с листингом  |
|  /C  | Создать файл перекрестных ссылок |

 Пример.

 \*=TEST/L/P оттранслировать файл TEST.MAC. Объектный файл выводить в файл TEST.REL, а листинг в файл TEST.LST.

**3.3. Формат листинга**

На каждой странице листинга первые две строки содержат:

 [TITLE ТЕКСТ] MACRO PAGE X [-Y]

 [SUBTITLE ТЕКСТ]

где TITLE ТЕКСТ - текст заданный псевдокомандой TITLE; X - основной номер страницы. Он увеличивается только при появлении из исходного файла символа подачи новой страницы. Когда печатается таблица символов X=S; Y - младший номер страницы. Он увеличивается при появлении псевдокоманды PAGE или при заполнении текущей страницы; SUBTITLE ТЕКСТ- текст, заданный псевдокомандой SUBTITLE. После вывода этих двух строк выводится одна пустая строка, а затем очередная строка программы. Строка программы имеет вид:

 [CRF#] [ERROR] LOC#M !XX!XXXX! ...... исходный текст

Если выводится таблица перекрестных ссылок, то первый элемент в строке - номер для таблицы, после него стоит символ табуляции.

ERROR - это флаг ошибки, если в строке есть ошибка. Если в строке нет ошибки, то вместо ERROR стоит пробел. Если таблица перекрестных ссылок не создается, то флаг ошибки стоит в первой колонке.

 LOG#M - значение счетчика адреса, которое в зависимости от

ключей /M или /0 представляет собой 16-е число, содержащее

4 цифры, или 8-е число, содержащее 6 цифр. Символ после числа -

это индикатор перемещаемости:

 ' программный перемещаемый;

 " данных перемещаемый;

 ! COMMON перемещаемый;

 ПРОБЕЛ абсолютный;

 \* внешний.

Далее печатаются три пробела, после которых идет код команды и операндов (если они есть у данной команды). Если строка листинга появилась из файла, включенного по псевдокоманде INCLUDE, то после кодов идет буква C. Если строка содержит макрорасширение (MACRO, REPT, IRP, IRPT), то после кодов стоит знак "+". Остаток строки содержит исходный текст.

В конце листинга выводится таблица символических имен. Сначала в алфавитном порядке печатаются имена всех макрокоманд, после этого выводятся все символические имена программы. После каждого символического имени выводится знак табуляции, а затем печатается значение символического имени. Если имя типа PUBLIC, то после его значения печатается буква I. Далее печатается один из следующих знаков:

 U имя не определено;

 C имя блока COMMON;

 \* внешнее имя;

 ПРОБЕЛ абсолютное имя;

 ' программный относительный;

 " данных относительный;

 ! COMMON относительный.

**3.4. Сообщения об ошибках**

Сообщения об ошибках индицируются одной буквой (флагом) в первой колонке листинга. Если файл с листингом не выводится на консоль, то ошибочные строки все равно выводятся.

|  |  |
| --- | --- |
| Флаг | Значение |
|  A  | Ошибка в аргументе. Неправильный формат аргумента псевдокоманды или аргумент вне допустимых границ. |
|   |
|   |
|  |  |
|  C  | Ошибка условного вложения. ELSE без IF; ENDIF без IF; два ELSE на один IF. |
|   |
|  |  |
|  D  | Многократно определенное символическое имя. |
|   |  |
|  E  | Ошибка при использовании EXTRN.Недопустимое использование символического имени, объявленного EXTRN. |
|   |
|   |
|  |  |
|  M  | Многократно определенное символическое имя. |
|  |  |
|  N  | Ошибка в числе. Неверное задание числа (например 8Q) |
|   |  |
|  |  |
|  O  | Неверный код операции или ошибочный синтаксис ENDM, LOCAL вне блока; SET, EQU или MACRO без имени; неверный синтаксис в коде команды; неверный синтаксис в выражении. |
|   |
|   |
|   |
|  |  |
|  P  | Ошибка фазы.Значение метки или имени в EQU изменилось при втором проходе. |
|   |
|   |
|  |  |
|  Q  | Вопрос.Обычно означает, что строка окончена неверно. Это предупреждение о возможной ошибке. |
|   |
|   |
|  |  |
|  R  | Перемещение.Недопустимое использование перемещения в выражении. |
|   |
|   |
|  U  | Неопределенное символическое имя. |
|  |  |
|  V  | Ошибка значения. |
|   | При первом проходе псевдокоманда не может получить необходимого определенного значения. Если символ, встретившийся в псевдокоманде, определен ниже, то при втором проходе ошибка U выдаваться не будет. |
|   |
|   |
|   |

 Кроме флагов компилятор выдает следующие сообщения об ошибках:

'NO END STATEMENT ENCOUNTED ON INPUT FILE '

Нет оператора END

'UNTERMINATED CONDITIONAL'

По меньшей мере один условный блок не завершен до конца файла

'UNTERMINATED REPT/IRP/IRPC/MACRO'

По меньшей мере один блок не завершен

[XX] [NO] FATAL ERROR(S) [,XX WARNINGS]

Количество серьезных ошибок и предупреждений.

Это сообщение всегда выводится на консоль и в файл с листингом

**4. Программа вывода перекрестных ссылок**

Для создания листинга с перекрестными ссылками сначала компилятор M80 должен создать специальный файл. MACRO создает этот файл, если задан ключ /C. При задании ключа /C вместо файла .LST компилятор создает файл типа .CRF. После того, как M80 закончит трансляцию, надо загрузить программу CREF.

После загрузки CREF выводит на экран запрос \* и ждет ввода командной строки. Формат командной строки для CREF:

 \* лист.файл = исх.файл

По умолчанию тип исходного файла - .CRF

После этого CREF создает файл типа .PRN, содержащий таблицу перекрестных ссылок.

Листинг такого файла отличается от стандартного следующим:

- каждое предложение начинается с номера;

- в конце листинга в алфавитном порядке печатаются все символические имена с номерами тех строк, где они встретились.

Номер строки, в которой данное символическое имя определено, отмечается знаком #.