ИНСТРУМЕНТЫ ОТЛАДКИ МАКРОСРЕДСТВ

1. Окно командной строки (директивы вывода сообщений препроцессора ECHO и %ECHO)
2. Листинг программы (машинные команды и макрорасширения макросов)
3. Директивы управления листингом (.LALL, .XALL .SALL, .NOLIST в MASM16 и те же или подобные в MASM32: .LISTALL, .LISTMACROALL) и составом макроопределений (INCLUDE, PUHGE).

Примеры их использования и дополнительные сведения о них будут вводиться далее по мере необходимости, а сейчас коротко: окно командной строки следует настроить на папку проекта и выполнять трансляцию командой

ml.exe /c /Fl Имя.asm

где ml.exe – программа, выполняющая препроцессорную обработку и затем – только компиляцию (ключ /c) файла Имя.asm в объектный модуль(Имя.obj) и построение файла листинга (Имя.lst).

МАКРОСРЕДСТВА MASM

ОСНОВНЫЕ ПОНЯТИЯ

1. **МАКРООПРЕДЕЛЛЛЕНИЕ** - специальным образом оформленная последовательность предложений языка ассемблера, под управлением которой ассемблер (точнее, его часть, называемая МАКРОГЕНЕРАТОРОМ или ПРЕПРОЦЕССОРОМ) порождает макрорасширения макрокоманд.
2. **МАКРОРАСШИРЕНИЕ** - последовательность предложений языка ассемблера (обыкновенных директив и команд), порождаемая макрогенератором при обработке макрокоманды под управлением макроопределения и вставляемая в исходный текст программы вместо макрокоманды.
3. **МАКРОКОМАНДА** (или **МАКРОВЫЗОВ**) - предложение в исходном тексте программы, которое воспринимается макрогенератором как команда (приказ), предписывающая построить макрорасширение и вставить его на ее место.

В макрокоманде могут присутствовать параметры, если они были описаны в макроопределении.

Макроопределение без параметров однозначно определяет текст макрорасширения.

Макроопределение с параметрами описывает множество (возможно, очень большое) возможных макрорасширений, а параметры, указанные в макро-команде, сужают это множество до одного единственного макрорасширения.

ДОСТОИНСТВА И НЕДОСТАТКИ МАКРОСРЕДСТВ ПО СРАВНЕНИЮ С ПОДПРОГРАММАМИ

Так как текст макрорасширения вставляется на место макрокоманды, то

+++ нет затрат времени, как для подпрограмм, на подготовку параметров, передачу управления и выполнение других работ при выполнении программы, но

--- при многочисленных вызовах МО (макроопределения) разрастается объем модуля программы,

--- фактические значения параметров макрокоманд должны быть известны препроцессору или могли быть вычислены им (нельзя использовать в качестве фактического параметра МО значения переменных или регистров, так как они могут быть известны только при выполнении программы).

Далее будут рассмотрены

1. Макроопределения (МО), макрокоманды (МК), макрорасширения.

2. Библиотеки МО. Управление составом МО.
3. Управление листингом.
4. Локализация имен в макроопределениях.
5. Условное ассемблирование в МО и вне их.
6. Генерация ошибок в МО и вне их.
7. Директива EXITM
8. Рекурсивные макроопределения.
9. Операции в макроопределениях.

БЛОКИ ПОВТОРЕНИЯ
---
1. Блок REPT.
2. Блок IRP или FOR
3. Блок IRPC или FORC
4. Блок WHILE

1. Макроопределения (МО), макрокоманды (МК), макрорасширения.

МО делят на однострочные и многострочные. Однострочные МО создаются в основном директивами = (равно), EQU (эквивалентно), TEXTEQU (эквивалентно тексту). Пояснения к ним будут приводиться по мере их использования в примерах.
Многострочные МО делят на макропроцедуры (МП) и макрофункции (МФ). Они имеют много общего в собственно определении, но принципиально различаются в конструкциях макрокоманд и в использовании директивы EXTERN. Поэтому сначала на примерах рассмотрим МП, а потом – коротко их отличия от МФ.

1.1 Структура макроопределений

```
ИмяМО MACRO СписФормПар     - заголовок МО
. . . . . .\
предложения языка ассемблера  > - тело  МО
. . . . . ./
       ENDM                 - конец текста МО
```

где
ИмяМО - имя макроопределения (МП и МФ),
СписФормПар - имена через запятую, используемые в предложениях тела макроопределения. СписФорПар может отсутствовать.
Завершение работы МФ должно происходить по директиве EXITM с параметром (значение которого будет представлять результат макрофункции – число или текст), а МП –при достижении директивы ENDM, или по директиве EXITM без параметра.

1.2 Формат макрокоманды макропроцедуры (КМП)

```
ИмяМП СписФактПар
```

Формат макрокоманды макрофункции (КМФ)

```
ИмяМФ(СписФактПар)
```

где
ИмяМО - имя МО, к которому происходит обращение,
СписФактПар - список фактических параметров через запятую, заменяющих при построении макрорасширения соответствующие формальные параметры. Элементами списка могут быть имена, строки, выражения.

Пример. МО для вывода на экран символа, заданного константой, или в регистре, или в ОП (в оперативной памяти)

```
PutCh   MACRO Ch      ;;Заголовок МО
```

```
        MOV DL,Ch ;;Ch содержит символ, который
        MOV AH,2  ;;будет выведен на экран
        INT 21H
    ENDM          ;;Конец МО
X1      DB      'AB'
        PutCh X1      ;*** -> MOV DL,X1    см. L8d.lst
        PutCh X1[1]   ;*** -> MOV DL,X1[1]
        PutCh 13      ;*** -> MOV DL,13
        PutCh 10      ;*** -> MOV DL,10
```

Получите листинг этого кода в MS DOS

Замечания.

! **Имена формальных** параметров МО-й **локализованы** в них, т.е. вне определения могут использоваться для обозначения других объектов.

! Число формальных параметров ограничено лишь длиной строки, обрабатываемой ассемблером.

! МО-я должны предшествовать обращениям к ним.

! Нет ограничений, кроме физических, на число предложений в теле МО.

! В листинге предложениям макрорасширений предшествуют ЦБЗ, указывающие глубину их вложения в макроопределениях.

2. Библиотеки МО. Управление составом МО.
Директивы INCLUDE, PURGE.

2.1 Директива INCLUDE

INCLUDE ФБиб

Функция: Включает в ассемблируемый текст текст из файла ФБиб на место директивы INCLUDE.

! В листинге предложения из подключенных файлов помечаются символом C. INCLUDE файл сам может содержать эту директиву.

Пример.

В файле LIB.2 содержится:

```
   INCLUDE LIB.1  - подключение текста из файла
                          LIB.1 (см. ниже) и
Scip   MACRO N      - МО пропуска N строк на экране
         MOV CX,N           со
MScip: NewStr     <------ ссылкой на МО из LIB.1
       loop MScip
       ENDM
```

В файле LIB.1 содержится:

```
NewStr MACRO       - МО пропуска одной строки на экране
       PutCh 13
       PutCh 10
       ENDM
```

Требуется вывести символы A и B разделенные 2-я пустыми строками: *

```
   INCLUDE LIB.2
        PutCh 'A'
        Scip  3    ;см. макрорасширения в файле листинга!
        PutCh 'B'
```

2.2 Директива PURGE

PURGE ИмяМО[,ИмяМО..]

Функция: Удаляет из обработки макроопределения, имена

которых перечислены в качестве параметров.
Цель - экономия памяти при работе ассемблера.

! MASM не генерирует ошибку, если есть макрокоманда, вызывающая удаленное макроопределение.

Пример.

```
INCLUDE LIB.2
PURGE Scip
     PutCh 'C'
     NewStr
     Scip  2    ;см. макрорасширения в файле листинга!
     PutCh 'D'
```

3. Управление листингом.

3.1 Директива .LALL: далее включать в листинг программы полные макрорасширения (кроме комментариев после ;;).

3.2 Директива .XALL (.LISTMACROALL): далее включать в листинг программы только те предложения макрорасширений, которые генерируют коды и данные.

3.3 Директива .SALL: далее не выводить тексты макрорасширений в листинг.

3.4 Директива .NOLIST: далее вообще прекратить вывод листинга до появления иной директивы.

Пример.

```
.LALL
  A3_1  MACRO
        NOP   ;  MO A3_1
        NOP   ;; MO A3_1
        ENDM

        A3_1
.SALL
  A3_2  MACRO
        NOP
        ENDM

        A3_2
```

4. Локализация имен в макроопределениях.

4.1 Директива LOCAL

Формат:

LOCAL Имя[,Имя..]

Функция: Имена переменных, меток, макроимена, перечисленные в LOCAL, будут локализованы в макроопределении.

Цель - использование одинаковых имен для обозначения разных объектов в разных МО-х модуля и вне их.

Пример.

```
.XALL
C4 EQU 0
  A4_1  MACRO
        LOCAL MScip,X4,C4
      JMP MScip
C4 EQU 6        ;   C4 – локальная константа,  а ??0002 - её новое имя
X4      DW   C ;   X4 - локальная переменная, а ??0001 - её новое имя
MScip:  MOV AX,X4 ;MScip - локальная метка  , а ??0000 - её новое имя
          ENDM      ;       !   см. Листинг
```

Comment ^ В макрорасширениях МО имена, объявленные локальными, будут автоматически заменены на не имеющие повторений имена вида ??nnnn, так что при многократных вызовах одного и того же МО не будет повторов локальных имён. ^

```
  A4_2  MACRO
        LOCAL MScip
      JMP MScip
MScip:  NOP       ;MScip - локальная метка, а ??0003 - её новое имя
        ENDM

X4      DW   0
        A4_1 C4     ;AX:=6
        A4_2
        JMP MIFB
```

5. Условное ассемблирование в МО и вне их.

Структуры условного ассемблирования IF W, где W, W1, W2,… вычисляемые препроцессором выражения:

```
  1)    IF.W.      Директива IF условного ассемблирования
        . . . . . \  Блок предложений ассемблера, выполняемых
        . . . . . /  при истинности условия
        ENDIF      Директива условного ассемблирования

  2)    IF.W1.      Директива IF условного ассемблирования
        . . . . . \  Блок1 предложений ассемблера, выполняемых
        . . . . . /  при истинности условия
        ELSEIF W2    Директива условного ассемблирования
        . . . . . \  Блок2 предложений ассемблера, выполняемых
        . . . . . /  , если условие ложно
        ELSE       Директива условного ассемблирования
        . . . . . \  Блок3 предложений ассемблера, выполняемых
        . . . . . /  , если условие ложно
        ENDIF      Директива условного ассемблирования
Блоков ELSEIF может быть несколько или не быть совсем.
```

Другие директивы IF условного ассемблирования:

```
  IFB  <par> - условие истинно, если фактический параметр par
               не был задан в МКоманде (скобки <> обязательны)
  IFNB <par> - условие истинно, если фактический параметр par
               был задан в МКоманде (скобки <> обязательны)
  IFIDN <s1>,<s2> - условие истинно, если строки s1 и s2
                    совпадают (скобки <> обязательны)
  IFDIF <s1>,<s2> - условие истинно, если строки s1 и s2
                    различаются  (скобки <> обязательны)
  IF  www     - условие истинно, если значение www<>0
  IFE www     - условие истинно, если значение www=0
  IFDEF  nam - условие истинно, если имя nam было описано выше
  IFNDEF nam - условие истинно, если имя nam не было описано выше
  IF1        - условие истинно 1-м шаге ассемблирования
  IF2        - условие истинно 2-м шаге ассемблирования
```

5.1 Директива IFB.

IFB <par>

Функция: Условие истинно, если фактический параметр par не был задан в МКоманде (макрокоманде) (скобки <> обязательны).

Цель - проверка наличия фактических параметров

5.2 Директива IFNB.

IFNB <par>

Функция: Условие истинно, если фактический параметр par был задан в МКоманде (скобки <> обязательны)

Цель - проверка наличия фактических параметров

Пример. МО вычисления суммы N элементов массива, адрес которого передается через BX. Результат сохраняется в R (по умолчанию - в регистре AX). Сохранить BX и AX, если AX не представляет результат.

МО можно располагать и внутри, и вне сегментов

```
SM      MACRO    N,R
   LOCAL M
   IFNB <R>
        PUSH AX
   ENDIF
        PUSH BX
        XOR AX,AX
        MOV CX,N
M:      ADD AL,[BX]
        INC BX
        LOOP M
        POP BX
   IFNB <R>
        MOV R,AX
        POP AX
   ENDIF
        ENDM
      ;в сегменте данных
REZ     DW  0
ARRW    LABEL WORD
ARR     DB  1,2,3,4,5,6
;в сегменте кода
MIFB:   LEA BX,ARR
        SM  6,DX ; см листинг
        SM  6    ; см листинг
        SM       ; см листинг
```

5.3 Директива IFIDN.

IFIDN <s1>,<s2>

Функция: Условие истинно, если строки s1 и s2 совпадают

Цель - проверка фактических параметров

5.4 Директива IFDIF.

IFDIF <s1>,<s2>

Функция: Условие истинно, если строки s1 и s2 не совпадают

Цель - проверка фактических параметров

Пример. МО вычисления суммы N элементов массива, адрес которого передается через BX, а результат сохраняется в R (по умолчанию - в регистре AX). Если

```
            первый параметр будет задан символом 0, то результат
            также должен быть равен 0                              *

SUM     MACRO    N,R
 IFIDN <0>,<N>
   IFB <R>
        MOV AX,0
   ELSE
        MOV R,0
   ENDIF
 ELSE
   SM   N,R
 ENDIF
        ENDM

         SUM 0;результат IFIDN <0>,<N> положительный
d  EQU 0 ;определение макроконстанты со значением 0
         SUM O ;результат IFIDN <0>,<N> отрицательный
```

5.5 Директива IF.

IF W

Функция: Условие истинно, если значение выражения W<>0.

5.6 Директива IFE W

IFE W

Функция: Условие ложно, если значение выражения www=0.

! Директивы IF и IFE могут использоваться для анализа выражений вне и внутри макроопределений, в том числе для анализа параметров.

Пример. МО вычисления суммы N элементов массива, адрес которого передается через BX, а результат сохраняется в R (по умолчанию - в регистре AX). Если первый параметр будет иметь значение <= 0, то результат должен быть равен 0

```
SUMM    MACRO    N,R
 IF N LE 0
   IFB <R>
        MOV AX,0
   ELSE
        MOV R,0
   ENDIF
 ELSE
   SM   N,R
 ENDIF
        ENDM

        SUMM 0
        SUMM O
        SUMM O-2,REZ
```

5.7 Директива IFDEF.

IFDEF nam

Функция: Условие истинно, если имя nam было определено ранее.

5.8 Директива IFNDEF

```
              IFNDEF  nam
```
Функция: Условие ложно, если имя nam не было описано выше.

! Директивы IFDEF и IFNDEF могут использоваться для анализа выражений вне и внутри макроопределений, в том числе для анализа параметров.

Пример. МО вычисления суммы N элементов массива, адрес которого передается через BX, а результат сохраняется в R (по умолчанию - в регистре AX). Если первый параметр будет иметь значение <= 0, то результат должен быть равен 0

```
NMIN = 3
    IFNDEF NMAX
      IFDEF NMIN
        SUMM  NMIN,REZ
      ELSE
        SUMM  6,REZ
      ENDIF
    ELSE
        SUMM  NMAX,REZ
    ENDIF
```

6. Генерация ошибок в МО и вне их.

Директива	№ ошибки	Текст сообщения на экране
.ERRB <s>	94	Forced error - пустая строка
.ERRNB <s>	95	Forced error - непустая строка
.ERRIDN <s1>,<s2>	96	Forced error - строки одинаковые
.ERRDIF <s1>,<s2>	97	Forced error - строки разные
.ERRE www	90	Forced error - выражение = 0
.ERRNZ www	91	Forced error - выражение <> 0
.ERRNDEF nam	92	Forced error - имя не определено
.ERRDEF nam	93	Forced error - имя определено
.ERR1	87	Forced error - pass 1
.ERR2	88	Forced error - pass 2
.ERR	89	Forced error

7. директива EXITM

Формат 1: EXITM

Функция: Закончить построение макропроцедуры.

Пример. МО вычисления суммы N элементов массива по условию предыдущего примера, но добавлен один параметр M - имя массива, в котором вычисляется сумма и генерация ошибок трансляции в следующих ситуациях:

1) в макрокоманде не указано имя массива,
2) в макрокоманде первые параметр имеет тип не BYTE.

```
SUMMA    MACRO N,R,M
     .ERRB <M> ;; не задан массив
     IF TYPE M NE 1
        .ERR ;тип массива не BYTE
        IF1
           %OUT ОШИБКА: тип массива не BYTE
```

```
       ENDIF
       EXITM
     ENDIF
     LEA BX,M
     SM  N,R
     ENDM

          SUMMA 6,,ARR
          SUMMA ,,ARR
          SUMMA  6,REZ,ARRW ;ОШИБКА: массив не BYTE
```

Формат 2: EXITM par

Функция: Закончить построение макрофункции и вернуть на место её вызова значение параметра **par**. Примеры см далее.

8. Рекурсивные макроопределения.

Пример макропроцедуры, вычисляющей S=P!.

```
MP_REC  MACRO P,S
      S=P
        MOV EAX,P
    IFE P EQ 1
            S=1
    ELSE
        MP_REC P-1
            S=S*P
    ENDIF
        ENDM
.CODE
     MP_REC 3,S   ;  ! см. Листинг ниже:
     MOV EAX,S
```

В листинге мы видим, что на каждом шаге рекурсии в макропеременной P будет меняться **текст** выражения, но значение вычисляться не будет:

```
 00000000                P1:
                         MPR 3,S
                       1
                       1                                IF 3 EQ 1
                       1                                      S =1
                       1                                ELSE
                       1                                      MPR 3-1,S
                       2
                       2                                IF 3-1 EQ 1
                       2                                      S =1
                       2                                ELSE
                       2                                      MPR 3-1-1,S
                       3
                       3                                IF 3-1-1 EQ 1
 = 00000001            3                                      S =1
                       3                                ELSE
                       3                                      MPR 3-1-1-1,S
                       3                                      S = S*3-1-1
                       3                                ENDIF
 = 00000002            2                                      S = S*3-1
                       2                                ENDIF
 = 00000006            1                                      S = S*3
                       1                                ENDIF
 00000000  B8 00000006          MOV EAX,S
```

Однако это не повлияло на результат подстановки S в код команды

MOV EAX,S, что в данном случае видно из самого кода (B8 00000006), а также из того, что выражения всёже вычислялись при каждом ее переопределении, что видно из строк первого столбца после знаков = (= 00000001, = 00000002, = 00000006). Тем не менее, полагаться только на листинг не всегда надежно и может потребовать много времени, если операнд не непосредственный или макропеременная получила значение в недоступном для нас месте.
Поэтому дополнительно, а иногда и в первую очередь лучше воспользоваться директивой %ECHO для вывода значения макроса. Но если при подстановке в команду или в директиву определения переменной макрос любого типа (text или number) всегда заменяется своим значением, то по %ECHO это справедливо только для типа text, а для типа number, как для S (числовые макропеременные, определённые директивой =, имеют тип number) необходимо выполнить преобразование к типу text. Проще всего это сделать стандартной функцией @CatStr(S) непосредственно в директиве %ECHO, сопроводив поясняющим текстом, например, так: **%ECHO '@CatStr(%S)' = @CatStr(%S).** Если вставить эту директиву в исходный код рядом с командой MOV EAX,S, то в окне командной строки (но не в листинге) мы увидис текст:
'@CatStr(%S)' = 6.
Объединить вывод по %ECHO можно с исходным текстом модуля, но не с листингом, используя ключ **/EP**. А если ещё и перенаправить в файл **name.EP:**

```
ml.exe /c /Fl /EP name.asm> name.EP
```

то этот файл можно будет просматривать на одной из вкладок VS C++.

9. Операции в макроопределениях.

1) **%** 2) **<>** 3) **&** 4) **!** 5) **;;**

9.1 **Операция %**

Формат: **% www**

Функция: Вычислить выражение перед представлением числа в символьной форме.

Пример переопределения макропроцедуры MP_REC (см. выше) добавлением оператора % для вычисления изменённого выражения при переходе к очередному шагу рекурсии

```
MP_REC MACRO P
       MOV AX,P
    IF P
       MP_REC  %(P-1) ;;перед записью в MPасш-ние вычислить P-1
    ENDIF
    ENDM

    MP_REC  3   ;  ! см. листинг
    MOV EAX,S
%ECHO '@CatStr(%S)' = @CatStr(%S)
```

Результат остался прежним (S равно 6), как видно из исходного текста, объединённого с текстом вывода по %ECHO '@CatStr(%S)' = @CatStr(%S) в файле P1.EP:

```
 .CODE
P1:

                                S =1

                                S =1
```

```
                                    S =1
                                    MPR %1-1,S
                                    S = S*1
                                    S = S*2
                                    S = S*3
MOV EAX,S
ECHO '@CatStr(%S)' = 6
```

Правда, исчезли детали алгоритма макроопределения, остались (пере)определения макропеременных (но без вычислений), макрокоманды, вывод по %ECHO и то, с чем в конечном счете должен работать компилятор.

Ещё пример на использование оператор % при вычислении N! но с использованием макрофункции:

```
      MFREC MACRO N
      %ECHO @CATSTR(N)
            IF N GT 1
                  EXITM %MFREC(%(N-1))*(N)
            ELSE
                  EXITM %1
            ENDIF
      ENDM
.CODE
M TEXTEQU %3
R TEXTEQU %MFREC(M) ; R TEXTEQU %MFREC(3)
MOV ECX, R ; MOV ECX, MFREC(3)
```

файле P1.EP мы увидим:

```
M TEXTEQU %3
ECHO 3
ECHO 2
ECHO 1
      EXITM %MFR(%(1-1))*(1)
R TEXTEQU %6
MOV ECX, 6
```

В файле P1.lst мы увидим:

```
= 3                     M TEXTEQU %3
                       1                          %ECHO @CATSTR(M)
                       1                                IF M GT 1
                       2                          %ECHO @CATSTR(2)
                       2                                IF 2 GT 1
                       3                          %ECHO @CATSTR(1)
                       3                                IF 1 GT 1
                       3                                      EXITM %MFR(%(1-1))*(1)
                       3                                ELSE
                       3                                      EXITM %1
                       2                                      EXITM %MFR(%(2-1))*(2)
                       1                                      EXITM %MFR(%(M-1))*(M)
 = 6                    R TEXTEQU %MFR(M)
 00000005  B9 00000006          MOV ECX, R
```

9.2 **Операция <>**

Формат : <txt>

Функция: Текст txt может содержать любые символы и рассматривается как одно целое. Если <txt> используется в качестве фактического параметра, то txt выносится в макрорасширение без изменений.

Пример. Использование макрокоманды в качестве фактического

параметра макрокоманды

```
WZ     MACRO REG,TELO
       LOCAL M1,M2,M3
       MOV CX,0
M1:    CMP REG,0
       LOOPNE M2
       JMP M3
M2:    TELO
       JMP M1
M3:
       ENDM

BLOK   MACRO  N
       MUL N    ;;DX:AX:=N*AX
       DEC N    ;;N:=N-1
       ENDM

      MOV BX,3  ; ВЫЧИСЛИТЬ ФАКТОРИАЛ ДЛЯ 3
      MOV AX,1  ; ЗДЕСЬ БУДЕТ РЕЗУЛЬТАТ 6 = 3!

                        WZ ebx,<BLOK ebx ; - это МакроКоманда>
```

9.3 Операция &

Формат : &par | par& | &par&

Функция: Приклеить текст параметра par к тому, что стоит слева | справа | справа и слева.

Пример

```
PRIMER1 MACRO I,J,K
        MOV   A&I,J&L
        INC   E&K&X
        ENDM

        PRIMER1 H,B,C; ! см. Листинг ниже:
00000000  8A E3         1        MOV   AH,BL
00000002  41            1        INC   ECX
```

9.4 Операция !

Формат : !C

Функция: Считать C символом, а не знаком операции.

Пример. **A!&B** – это текст **A&B**, а не склейка B с A!.

9.5 Операция ;;

Формат : **;; txt**

Функция: Текст txt не выносится в макрорасширение.

БЛОКИ ПОВТОРЕНИЯ

1. Блок REPT.
2. Блок IRP или FOR.
3. Блок IRPC или FORC
3. Блок WHILE.

1. Блок REPT.

Структура блока

```
        REPT www                    - заголовок блока
        . . . . . . . . . . . . . .\
        предложения языка ассемблера  > - тело блока
        . . . . . . . . . . . . . ./
        ENDM                        - конец текста блока
```

где

www – выражение, значением которого является ЦБЗ, определяющее число повторных копирований в макрорасширение тала блока

Пример. Целочисленное деление содержимого BX на 8 *

```
MOV BX,16
REPT 3     ;BX:=BX div 8
   SHR BX,1
ENDM
```

Пример. Резервирование 3-х байтов с начальными значениями 0, 3, 6

```
  A=0
MB0 LABEL BYTE
  REPT 3
    DB A
     A=A+3
  ENDM
```

2. Блок IRP или FOR.

Структура блока

```
        IRP form,<fact_1[,fact_2,...]>  - заголовок блока
        . . . . . . . . . . . . . .\
        предложения языка ассемблера  > - тело блока
        . . . . . . . . . . . . . ./
        ENDM                        - конец текста блока
```

где

form – формальный параметр, используемый в теле блока

fact_i – фактический параметр (i=1,2,...), подставляемый на место form при i-м копировании тела блока в макрорасширение.

Пример. Определение переменных A0,A1,A2,A3 с начальными значениями 0,1,2,3 соответственно. *

```
IRP X,<0,1,2,3> ;параметры - числа
   A&X DB X
ENDM
```

Пример. Описание переменных Z_AAAA, Z_BBBB, Z_CCCC с начальными значениями 'AAAA', 'BBBB', 'CCCC' соответственно. *

```
DCL MACRO Y
  IRP X,<AAAA,BBBB,CCCC> ;параметры - строки символов
       Y&&X  DB '&X&'
    ENDM
ENDM

MIRP:    DCL Z_
```

3. Блок IRPC или FORC.

Структура блока

```
IRPC form,fact              - заголовок блока
```

```
      . . . . . . . . . . . . . . . .\
      предложения языка ассемблера  > - тело блока
      . . . . . . . . . . . . . . . ./
   ENDM                          - конец текста блока
   где
      form - формальный параметр, используемый в  теле блока
      fact - фактический параметр - строка символов, i-й
             символ которого (i=1,2,...)  подставляется  на
             место  form  при   i-м  копировании  тела блока
             в макрорасширение.
```

Пример. Описание переменных полей данных с начальными значениями 'A', 'B', 'C' соответственно.

```
00000000                .DATA
                                  IRPC X,<" 2 ABC">
                                     DB '&X&'
                                  ENDM
 00000000  22                  1               DB '"'
 00000001  20                  1               DB ' '
 00000002  32                  1               DB '2'
 00000003  20                  1               DB ' '
 00000004  41                  1               DB 'A'
 00000005  42                  1               DB 'B'
 00000006  43                  1               DB 'C'
 00000007  22                  1               DB '"'
```

Блок повторения WHILE

```
   Структура блока
      while w                - заголовок блока
      . . . . . . . . . . . . . . . .\
      предложения языка ассемблера  > - тело блока
      . . . . . . . . . . . . . . . ./
      ENDM                           - конец текста блока
   где
      w - выражение
```

Работа блока будет продолжаться, пока выражение w будет иметь значение истина (не ноль).

Пример. Стандартные макрофункция @SubStr и директива SubStr могут порождать множество подстрок типа text с числовыми и нечисловыми значениями, причём при одних и тех же значениях параметров директива SubStr определит (переопределит) макропеременную типа text, а макрофункция @SubStr вернёт значение, совпадающее со значением макропеременной. Следующий вложенный цикл позволяет перебрать и вывести значения подмножеств строки 1234

```
j=1
while j LE 4
      i=1
      WHILE i le 5-j
            names SubStr <ABCD>,i,j
            %ECHO 'names SubStr <ABCD>,i,j' out: names , i, j
            %ECHO '@SubStr (ABCD,i,j)' out: @SubStr (ABCD,i,j)
            i=i+1
      endM
      j=j+1
endm
```

Этот код представляет вывод всех подстрок строки ABCD (A, B, C,D,AB, BC, CD, ABC, BCD, ABCD), последовательно присваиваемых директивой макропеременной и возвращаемых макрофункциеей при изменяющихся значениях второго параметра (i) – начала подстроки и третьего параметра (j) – длина подстроки. Следующие строки представляют результаты вывода по %ECHO:

```
'names SubStr <ABCD>,i,j' out: A , 1,1
     '@SubStr (ABCD,i,j)' out: A
'names SubStr <ABCD>,i,j' out: B , 2,1
     '@SubStr (ABCD,i,j)' out: B
'names SubStr <ABCD>,i,j' out: C , 3,1
     '@SubStr (ABCD,i,j)' out: C
'names SubStr <ABCD>,i,j' out: D , 4,1
     '@SubStr (ABCD,i,j)' out: D
'names SubStr <ABCD>,i,j' out: AB , 1,2
     '@SubStr (ABCD,i,j)' out: AB
'names SubStr <ABCD>,i,j' out: BC , 2,2
     '@SubStr (ABCD,i,j)' out: BC
'names SubStr <ABCD>,i,j' out: CD , 3,2
     '@SubStr (ABCD,i,j)' out: CD
'names SubStr <ABCD>,i,j' out: ABC , 1,3
     '@SubStr (ABCD,i,j)' out: ABC
'names SubStr <ABCD>,i,j' out: BCD , 2,3
     '@SubStr (ABCD,i,j)' out: BCD
'names SubStr <ABCD>,i,j' out: ABCD , 1,4
     '@SubStr (ABCD,i,j)' out: ABCD
```