теория

1. Что может быть фактическим параметром макрокоманды

   любой текст (в том, числе пустой), сбалансированный по кавычкам и угловым скобкам, причём этот текст не должен содержать запятых и точек с запятой вне вышеуказанных кавычек и скобок.

2. что происходит с параметрами при макровызове

   при макровызове макропроцессор (=макрогенератор) находит макроопределение и подставляет в макрос фактические параметры вместо формальных, генерируя, таким образом, макрорасширение. При ответе на этот пункт нужна ещё такая добавка: при подстановке фактического параметра производится выполнение макрооператоров (&,<>,%,!)

3. После передачи параметров начинается просмотр тела макроопределения и поиск в нём предложений, содержащих макросредства, например, макрокоманд. Все предложения в макроопределении, содержащие макросредства, обрабатываются Макропроцессором так, что в результате получается набор предложений на «чистом» языке Ассемблера (уже без макросредств), такой набор предложений называется **макрорасширением** Последним шагом в обработке макрокоманды является подстановка полученного макрорасширения на место макрокоманды, это действие называется **макроподстановкой**

4. %: посчитает макровыражение пример:

```
.code                                   вывод:
    myMacro macro op1:req               M
        echo op1                        22
    endm                                N
Start:                                  mystring
    M equ (3*5+7)                       14
    N equ mystring                      7+7
    myMacro M
    myMacro %M
    myMacro N
    myMacro %N
    myMacro %7+7
    myMacro 7+7
    exit
end Start
```

5. ! - символ экранирования

```
.code                                   вывод
    myMacro macro op1:req               HelloWorld42
        echo op1                        HelloWorld%M
    endm                                HelloWorld!%M
Start:                                  Helloworld!
    M equ 7*6                           Helloworld
    myMacro HelloWorld%M                test.asm(17) : error A2038: missing operand for macro
    myMacro HelloWorld!%M               operator
    myMacro HelloWorld!!!%M             myMacro(1): Macro Called From
    myMacro Helloworld!!                test.asm(17): Main Line Code
    myMacro !Helloworld                 Helloworld!
    myMacro Helloworld!
    exit
end Start
```

   отметим что % и ! поставленные в конце выражения выдадут ошибку, необходимо экранировать

6. Если фактический параметр заключён в угловые скобки (<>), то эти скобки считаются макрооператорами выделения, они заставляют рассматривать заключённый в них текст как единое целое, даже если в нём

встречаются служебные символы. Обработка макровыделения заключается в том, что внешние парные угловые скобки отбрасываются при передаче фактического параметра на место формального. Обратите внимание, что отбрасывается только одна (внешняя) пара угловых скобок, если внутри фактического параметра есть ещё угловые скобки, то они сохраняются.

```
.code                                                   вывод
    myMacro macro op1:req, op2:=<default>               HelloWorld42, I‘m
        echo op1                                        default
        echo op2                                        HelloWorld42
    endm                                                default
Start:                                                  HelloWorld42
    SEMICOLON_CHAR TEXTEQU <;>                          I‘m
    M equ 7*6
    myMacro <HelloWorld%M, I‘m>
    myMacro <HelloWorld%M ; I‘m> ; внутри комментарий
    myMacro HelloWorld%M, I‘m
    exit
end Start
```

7. & - разделитель лексемы! важно могло слоиться неправильное впечатление что это указатель или разыменователь но это не так то что необходимо подставить надо выделять! пример

```
M1 macro x
    for i,<1,2,3>
        x&&i db i
    endm
endm
.data
    M1 myvar
    ; сгенерит myvar1 db 1 и другие
```

с двух сторон отделить то что подставляешь за исключением конца и начала строки там только с одной стороны если у нас подстановка из одного имени то & не нужен

```
M1 macro x, ze
    for i,<1,2,3>
        x&U&ze&&i&q db ’$i$’
    endm
endm
.data
    M1 myvar, _
    ; сгенерит myvarU_1q db ’1’ и другие
```

иной пример

что будет если
если записать x&i то i не подставится: myvari
если записать x&U&ze&&i&q но макропеременной ze нет: myvarU&ze&1q
если записать x&U&ze&i&q: myvarU&ze1q

табличка конструкций

type	размер одно элемента в байтах если константа то 0,
size	размер элементов до первой запятой size real8 6 dup (?) = 48
sizeof	размер элементов включая все перечисления через запятую
length	число элементов до запятой то есть вернёт 1 или то что перед dup
lengthof	число всех элементов после метки

директивы условной компиляции

ifb <op1>	(if blank) если параметр пустой то подстановку выполняем
ifnb <op1>	(if not blank)
ifdif <op1>, <op2>	(if difference) если разные то выполняем
ifidn <op1>, <op2>	(if identical) если одинаковые то выполняем
ifdifi <op1>, <op2>	(if difference ignore case) если разные регистронезависимо ('string' = 'StRiNg') то выполняем
ifidni <op1>, <op2>	(if identical ignore case) если одинаковые регистронезависимо то выполняем

Директива If выполняет подстановку согласно булевым значениям в макросах всё что не 0 это true, 0 = false
логические выражения

EQ вместо = GE вместо ≥ GT вместо >
LE вместо ≤ LT вместо < NE вместо ≠

любой обозначенный 'if' завершается endif внутри могут быть выражения: else и elseif[idn|idni|dif|...]
операции в макровыражениях и их приоритет:

1. (), [], <>, length[of], size[of]
2. . (точка)
3. : (переопределение сегмента по умолчанию)
4. ptr, offset, seg, type, this
5. high, low, highword, lowword (возвращает старшую и младшую половины от слова и от двух слов соответственно)
6. Одноместные (унарные) + и − (для корректности 6+-+7)
7. *, / (в смысле div), mod, shl, shr
8. бинарные + и −
9. EQ, NE, LT, LE, GT, GE
10. not
11. and
12. or, xor
13. short, .type

переходы внутри определния можно выполнять по макрометкам
перед их использованием необходимо их объявить в начале макроса прописав: local L
после чего где то внутри макроса поставить метку:
:L (в отличии от меток asm : ставится перед), переход осуществляется по goto L

также директива local для Макропроцессора имеет следующий смысл. При каждом входе в макроопределение локальные имена, перечисленные в этой директиве, получают новые уникальные значения. Обычно Макропроцессор выполняет это совсем просто: при первом входе в макроопределение заменяет локальное имя L на имя ??0001
для преждевременного выхода из макроса используется .exitm
благодаря этой директиве можно заранее выйти из цикла или из всего макроса в случае ошибки при макропроцессировании
.err <сообщение> - породит ошибку при компиляции
opattr операнд - возвращает 16 байт:

?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

; Bit Set If...
; 0 имя метки процеду
; 1 перемещаемая переменная (область данных)
; 2 константа
; 3 перемещаемый адрес
; 4 регистровое выражение
; 5 определён
; 6 операнд в стеке (usually a LOCAL variable or parameter)
; 7 имя в extrn label
; 8-10 Language type (0=no type)
что другие да фиг пойми вроде соглашения и чёт зарезервировано не нагуглил
первые 8 бит аналогичны .type
просто по циклам пример отрывок макроса проверить на регистр что можно менять по соглашению eax, edx, ecx

```
local K
K=0; это var K: integer:=0
for reg,<eax, edx, ecx, ax, dx, cx, al, ah, dl, dh, cl, ch>
    ifidni <reg>,<X>
        K=1; это K:=1
        exitm; преждевременный выход из цикла
    endif
endm
```

по циклу forc

```
local K
K=0; это var K: integer:=0
forc reg,<adc>
    ifidni <reg>,<e&reg&x>; проверка лишь на 4 байтные
        K=1; это K:=1
        exitm; преждевременный выход из цикла
    endif
endm
```

цикл **while** выражение как в if - если что метки есть
цикл **repeat** N