数据定义

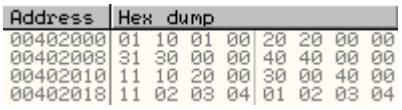
数据类型

MASM以数据所占内存字节数定义了多种数据类型

- BYTE，db，8位=1字节
- WORD，dw， 16位=2字节
- DWORD，dd， 32位=4字节（32个二进制位，8个16进制字符）
- QWORD ，dq，64位=8字节

```
array DWORD 11001h,2020h
      DWORD 3031h,4040h
barry WORD 1011h,20h
      WORD 30h,40h
carry BYTE 11h,2h,
           3h,4h
darry BYTE 1h,2h
      BYTE 3h,4h
```

内存存储如下：



初始值：数据定义语句中要指定至少一个初始值，即使这个初始值是0。如果有多个初始值的话，那么应以逗号分隔。对于整数数据类型，其初始值可以是与变量的数据类型(BYTE,WORD等)尺寸相匹配的整数常量或表达式。如果在定义中不想初始化变量(赋予变量一个随机值)，那么可以使用符号“?”作为初始值，可放在.data?段。所有的初始值，不管其格式如何，均由编译转换为二进制数
据。比如00110010b，32h和50d都将产生同样的二进制值、原因即在于此。

可以指定多个初始值，多个初始值用逗号隔开，如 `my_var DWORD 0, 1, 2, 3`

定义字符串

```
str_hello BYTE "Hello World!", 0Dh, 0Ah
          BYTE "I love assembly language"
          BYTE 0Dh, 0Ah, 0
```

0Dh和0Ah是CR/LF（回车、换行）的ASCII编码

字符串的结尾是0

```
Address  Hex dump                  ASCII
00402000 01 F0 01 00 20 20 00 00 ☺?.  ..
00402008 48 65 6C 6C 6F 20 57 6F Hello Wo
00402010 72 6C 64 21 0D 0A 49 20 rld!..I
00402018 6C 6F 76 65 20 61 73 73 love ass
00402020 65 6D 62 6C 79 20 6C 61 embly la
00402028 6E 67 75 61 67 65 0D 0A nguage..
```

DUP操作符

DUP操作符使用一个常量表达式作为计数器为多个数据项分配储存空间，可用于为字符串和数组分配空间。

```
BYTE 20 DUP(0)   ;20字节，全等于0
BYTE  4 DUP("STACK") ;20字节:"STACKSTACKSTACKSTACK"
```

字数组

字数组：可以通过显式初始化每个元素或使用 DUP 操作符创建字数组，下面是一个包含特定初始值的字数组的例子：

```
myList  WORD 1,2,3,4,5
```

下面是该数组在内存中的图解，图中假设 myList 从偏移 0000 处开始，注意地址是以 2 递增的，因为每个元素值占用两个字节：

偏移	值
0000:	1
0002:	2
0004:	3
0006:	4
0008:	5

DUP 操作符为初始化多个字提供了方便：

```
array WORD 5 DUP(?)          ; 5 个未初始化的值
```

符号常量

符号常量是通过将标识符与整数表达式或文本联系起来而创建的。与保留储存空间的变量不同，符号常量并不占用任何实际的储存空间。符号常量仅在编译期间汇编器扫描程序时使用，运行期间不能更改。

等号伪指令

等号伪指令，将符号名和整数表达式联系起来。通常，表达式是32位的整数值，汇编程序的时候，所有出现名字的地方都由汇编器在预处理阶段替换为对应表达式的值。

```
COUNT = 500
mov eax,  COUNT
arry COUNT DUP(0)
```

计算数组和字符串的大小

MASM用$运算符存储当前语句的地址偏移。$可以用来计算数组或字符串的大小。

```
myString Byte "acbd"
myString_len = ($-myString)
list Byte 10,20,30,40
listsize = ($-list)  ;必须紧跟
dw_array DWORD 0, 1, 2, 3, 4
array_size = ($ - dw_array)/4
```

EQU伪指令

EQU伪指令将符号名与整数表达式或任意文本联系起来。不允许重定义。

EQU 伪指令将符号名同整数表达式或任意文本联系起来，有以下三种格式：

```
name EQU expression
name EQU symbol
name EQU <text>
```

在第一种格式中，表达式（expression）必须是有效的整数表达式（参见 3.1.2 节）；在第二种格式中，符号（symbol）必须是已用“=”或 EQU 定义的符号名；第三种格式中，尖括号内可以是任意文本。当汇编器在后面遇到已定义的“名字”（name）时，就用该名字代表的整数值或文本替代。当定义任何非整数的值的时候，EQU 就可能非常有用了，例如实数常量就可以用 EQU 定义：

```
PI EQU <3.1416>
```

例子：下例把一个符号同一个字符串联系了起来，然后使用该符号创建了一个变量：

```
pressKey EQU <"Press any key to continue...",0>
.
.
.data
prompt  BYTE   pressKey
```

一段程序

```
.386
.model flat, stdcall
option casemap:none
include D:\masm32\include\windows.inc
include D:\masm32\include\kernel32.inc
include D:\masm32\include\masm32.inc
includelib D:\masm32\lib\kernel32.lib
includelib D:\masm32\lib\masm32.lib
.data
BYTE  4 DUP("STACK")
array DWORD 20 DUP(1)
myString Byte "acbd"
mylen = ($-myString)
lengths byte mylen
a DWORD 30313235h
```

```
b word 3139h
cc byte 32h
dddd dword ?
eeee byte ?
ffff byte ?
.code
start:
mov lengths,mylen
mov eax,a
mov bx,b
mov cl,cc
mov dddd,eax
mov eeee,al
mov ffff,bl
invoke StdOut, addr dddd
end start
```

输出：521059

Address	Hex dump	ASCII
00403000	53 54 41 43 4B 53 54 41	STACKSTA
00403008	43 4B 53 54 41 43 4B 53	CKSTACKS
00403010	54 41 43 4B 01 00 00 00	TACK0...
00403018	01 00 00 00 01 00 00 00	0...0...
00403020	01 00 00 00 01 00 00 00	0...0...
00403028	01 00 00 00 01 00 00 00	0...0...
00403030	01 00 00 00 01 00 00 00	0...0...
00403038	01 00 00 00 01 00 00 00	0...0...
00403040	01 00 00 00 01 00 00 00	0...0...
00403048	01 00 00 00 01 00 00 00	0...0...
00403050	01 00 00 00 01 00 00 00	0...0...
00403058	01 00 00 00 01 00 00 00	0...0...
00403060	01 00 00 00 61 63 62 64	0...acbd
00403068	04 35 32 31 30 39 31 32	♦5210912

运行后：

Address	Hex dump	ASCII
00403000	53 54 41 43 4B 53 54 41	STACKSTA
00403008	43 4B 53 54 41 43 4B 53	CKSTACKS
00403010	54 41 43 4B 01 00 00 00	TACK0...
00403018	01 00 00 00 01 00 00 00	0...0...
00403020	01 00 00 00 01 00 00 00	0...0...
00403028	01 00 00 00 01 00 00 00	0...0...
00403030	01 00 00 00 01 00 00 00	0...0...
00403038	01 00 00 00 01 00 00 00	0...0...
00403040	01 00 00 00 01 00 00 00	0...0...
00403048	01 00 00 00 01 00 00 00	0...0...
00403050	01 00 00 00 01 00 00 00	0...0...
00403058	01 00 00 00 01 00 00 00	0...0...
00403060	01 00 00 00 61 63 62 64	0...acbd
00403068	04 35 32 31 30 39 31 32	♦5210912
00403070	35 32 31 30 35 39 00 00	521059..
00403078	00 00 00 00 00 00 00 00	....
00403080	00 00 00 00 00 00 00 00	....