目 录

1 结构

1.1 定义结构

结构使用 STRUCT 和 ENDS 伪指令定义，格式如下：

```
名字 STRUCT
    域的声明
名字 ENDS
```

如果结构的域有初始值，在定义结构变量时这些初始值就成了结构变量域的默认值。结构中可使用多种类型的初始值，有字符串、整数、数组。初始化数组时，可以使用 DUP 操作符初始化数组元素。还可以使用“?”，用于代表域未定义。以下是一个例子：

```
Employee STRUCT
    IdNum BYTE "00000000"
    LastName BYTE 30 DUP(0)
    FirstName BYTE ?
    SalaryHistory DWORD 0, 0, 0, 0
Employee ENDS
```

1.1.1 结构中域的对齐

通过 ALIGN 伪指令设置一个域或变量的地址对齐方式，格式如下：

```
ALIGN datatype
```

如果想让 myVar 对齐在双字地址边界上，如下操作：

```
ALIGN DWORD
myvar DWORD ?
```

以下是数据类型的地址对齐方式：

成员类型	对齐方式
BYTE, SBYTE	对齐在 8 位（字节）边界上
WORD, SWORD	对齐在 16 位（字）边界上
DWORD, SDWORD	对齐在 32 位（双字）边界上
QWORD	对齐在 64 位（8 字节）边界上
REAL4	对齐在 32 位（双字）边界上
REAL8	对齐在 64 位（8 字节）边界上
结构成员	其成员所要求的最大的对齐方式
联合成员	第一个成员要求的对齐方式

在结构中使用 ALIGN 伪指令使得成员 years 对齐在字边界上、成员 SalaryHistory 对齐在双字边界上，如下所示：

```
Employee STRUCT
    IdNum BYTE "00000000"
    LastName BYTE 30 DUP(0)
    ALIGN WORD
    Years WORD 0
    ALIGN DWORD
    SalaryHistory DWORD 0,0,0,0
Employee ENDS
```

1.2 声明结构变量

声明结构变量的格式如下：

```
; identifier是标志符名
; structureType是已经定义的结构
identifier structureType <initializer-list>
```

初始值列表中的值按照从左到右的顺序对结构的相应成员依次赋值，可以插入逗号作为占位符跳过对结构中某些域的初始化，如下所示：

```
point1 COORD <5,10>
person3 Employee <,"dJones">
```

对于结构中数组类型的域，可以使用 DUP 操作符初始化某些或全部数组元素。如果初始化值比域短，那么剩余位置将用 0 填充。例子如下：

```
person4 Employee <,,,2 DUP(20000)>
```

对于结构中字符串类型的域，如果初始化值比域短，那么剩余位置以空格填充，字符串的末尾不会自动插入空字符。

1.3 结构中域成员的引用

汇编中对结构成员的引用和 C 语言类似，例子如下：

```
Employee STRUCT
    IdNum BYTE "000000000"
    LastName BYTE 30 DUP(0)
    ALIGN WORD
    Years WORD 0
    ALIGN DWORD
    SalaryHistory DWORD 0, 0, 0, 0
Employee ENDS

.data
worker Employee <>

.code
```

```
    mov dx, worker.Years
    mov worker.SalaryHistory, 20000
    mov [woker.SalaryHistory+4], 30000
```

1.4 间接寻址结构数据

使用间接操作数饮用结构时要求使用 PTR 结构符，如下所示：

```
    mov esi, OFFSET worker
    mov ax, (Employee PTR [esi]).Years
```

1.5 变址寻址结构数据

可以使用变址操作数访问结构数组。假设 department 是一个包含 5 歌 Employee 对象的数组，可以使用下面语句访问其索引位置 1 处的 Employee 对象的 Years 域：

```
    .data
    department Employee 5 DUP(<>)

    .code
    mov esi, TYPE Employee
    mov department[esi].Years, 4
```

2 联合

结构的每个域都有一个相对于结构第一个字节的偏移值，而联合中的所有域都是从同一偏移地址开始的，所以联合的大小等于其中最长的域的长度。联合使用 UNION 和 ENDS 伪指令声明。当联合不是某个结构的成员时，定义格式如下所示：

```
    unionname UNION
        union-fields
    unionname ENDS
```

当联合嵌套在结构中时，则格式如下：

```
    structname STRUCT
        ; ...
        UNION unionname
            union-fields
        ENDS
    structname ENDS
```

联合中所有域只能有一个初始化值，如下所示：

```
Integer UNION
    D DWORD 1
    W WORD 5
    B BYTE 8
Integer ENDS

.data
myInt Integer <>
```

myInt.D、myInt.W 和 myInt.B 都等于 1，而忽略了 W 和 B 的初始值。

除了所有域只能有一个初始化值这一点之外，联合的语法和结构的语法完全相同。

联合因为可以使用不同大小的操作数，所以在某些场合下具有灵活性，如下所示：

```
.data
val3 Integer <>

.code
mov val3.B, al
mov val3.W, ax
mov val3.D, eax
```

使用联合时同一时刻只能使用联合中的一种数据类型，以免造成数据的冲突。

3 宏