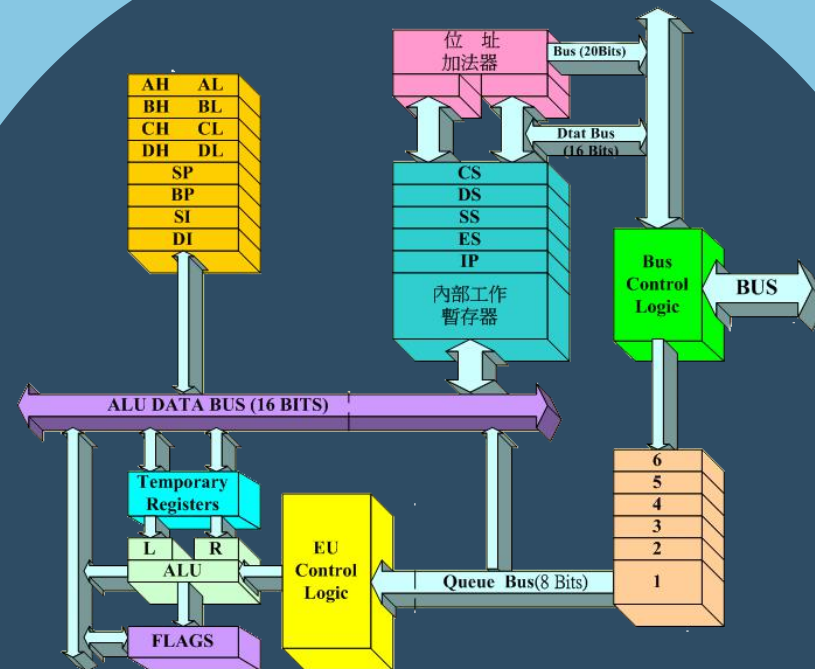


汇编伪操作汇总

贺利坚 主讲



汇编语言程序设计
Assembly Language

操作？伪操作？

```
assume .....  
.....  
code segment  
main proc far  
start:  
    mov ax, 0  
    add ax, bx  
    .....  
    mov ax, 4c00h  
    int 21h  
main endp  
code ends  
end start
```

- ❑ 汇编指令：对应机器指令，在程序运行期间由计算机执行。
- ❑ 伪操作：在汇编程序对源程序汇编期间，由汇编程序处理的操作，可以完成如数据定义、分配存储区、指示程序结束等功能。

伪操作

处理器选择伪操作

段定义伪操作

程序开始和结束伪操作

数据定义及存储器分配伪操作

表达式赋值伪操作

地址计数器与对准伪操作

基数控制伪操作

.8086 选择 8086 指令系统

.286 选择 80286 指令系统

.286P 选择保护模式下的 80286 指令系统

.386 选择 80386 指令系统

.386P 选择保护模式下的 80386 指令系统


.486 选择 80486 指令系统

.486P 选择保护模式下的 80486 指令系统

.586 选择 Pentium 指令系统

.586P 选择保护模式下的 Pentium 指令系统

伪操作


 处理器选择伪操作


 段定义伪操作

 程序开始和结束伪操作

 数据定义及存储器分配伪操作

 表达式赋值伪操作

 地址计数器与对准伪操作

 基数控制伪操作

```
assume cs:code, ds:data, es:extra
data segment      ; 定义数据段
    ...
data ends
;-----
extra segment     ; 定义附加段
    ...
extra ends
;-----
code segment      ; 定义代码段
start:
    mov ax, data
    mov ds, ax    ; 段地址    段寄存器
    ...
code ends
end start
```

段定义伪操作

ASSUME 段寄存器:段名[, 其他段声明]

段名 SEGMENT [定位类型] [组合类型] [使用类型] ['类别']

.....

.....

; 语句序列

段名 ENDS

定位类型 align_type

PARA BYTE WORD DWORD PAGE

组合类型 combine_type

PRIVATE PUBLIC COMMON STACK AT exp

使用类型 use_type

USE16 USE32

类别 'class'

完整的段定义伪操作

.MODEL 存储模式 [,其他选项]

存储模式：

- tiny
- small
- medium
- compact
- large
- huge
- flat

定义存储模式，指定在内存中如何安放各段。

.code [name]

.data

.data?

.fardata [name]

.fardata? [name]

.const

.stack [size]

简化的段定义伪操作

简写的 Hello world!



```
assume cs:codesg, ss:stacksg, ds:datasg
datasg segment
    str db 'hello world!$'
datasg ends
stacksg segment
    db 32 dup (0)
stacksg ends
codesg segment
start: mov ax,datasg
      mov ds,ax
      mov ax, stacksg
      mov ss, ax
      mov sp, 20h
      lea bx,str
output:mov dl, [bx]
      cmp dl, '$'
      je stop
      mov ah, 02H
      int 21h
      inc bx
      jmp output
stop: mov ax,4c00h
      int 21h
codesg ends
end start
```

```
.8086
.MODEL small
.data
    str db 'hello world!$'
.stack 20H
.code
start: mov ax,@data
      mov ds,ax
      lea bx,str
output:mov dl, [bx]
      cmp dl, '$'
      je stop
      mov ah, 02H
      int 21h
      inc bx
      jmp output
stop: mov ax,4c00h
      int 21h
end start
```

伪操作

🖥️ 处理器选择伪操作

🖥️ 段定义伪操作

🖥️ 程序开始和结束伪操作

🖥️ 数据定义及存储器分配伪操作

🖥️ 表达式赋值伪操作

🖥️ 地址计数器与对准伪操作

🖥️ 基数控制伪操作

```
TITLE text
NAME module_name
END [label]
.STARTUP
.EXIT [return_value]
```

```
.model small
.stack 100H
.data
.....
.code
.startup
.....
.exit 0
end
```

```
.model small
.data
.....
.code
start:
    mov ax, @data
    mov ds, ax
.....
    mov ax, 4c00h
    int 21h
end start
```

注意：MASM5.0/5.1不支持.startup和.exit

伪操作

🖥️ 处理器选择伪操作

🖥️ 段定义伪操作

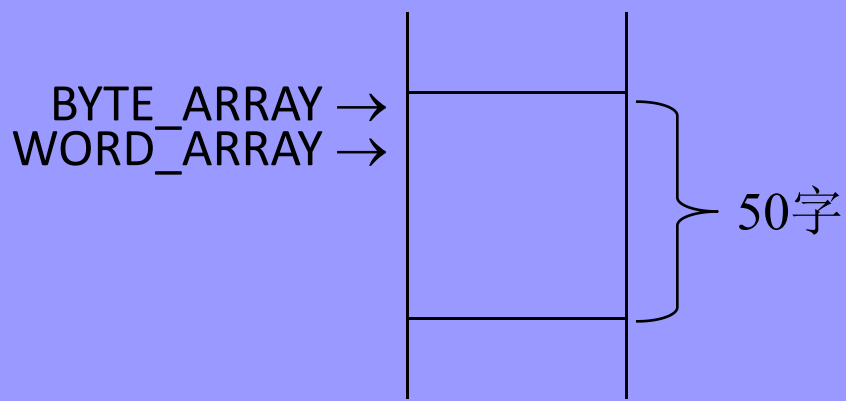
🖥️ 程序开始和结束伪操作

🖥️ 数据定义及存储器分配伪操作

🖥️ 表达式赋值伪操作

🖥️ 地址计数器与对准伪操作

🖥️ 基数控制伪操作



[变量]助记符 操作数 [,操作数, ...] [;注释]

助记符: DB DW DD DF DQ DT

```
DATA_BYTE    DB    10,4,10H,?
DATA_WORD    DW    100,100H,-5,?
              PAR1   DW    100,200
              PAR2   DW    300,400
ADDR_TABLE   DW    PAR1,PAR2
VAR          DB    100 DUP (?)
              DB    2 DUP (0,2 DUP(1,2),3)
```

变量名 LABEL type

LABEL 伪操作

功能: 同一变量 (同一空间) 将具有不同的类型

```
BYTE_ARRAY   LABEL   BYTE
WORD_ARRAY   DW    50  DUP (?)
```


伪操作

🖥️ 处理器选择伪操作

🖥️ 段定义伪操作

🖥️ 程序开始和结束伪操作

🖥️ 数据定义及存储器分配伪操作

🖥️ 表达式赋值伪操作

🖥️ 地址计数器与对准伪操作

🖥️ 基数控制伪操作

- 表达式名 EQU 表达式
ALPHA EQU 9
BETA EQU ALPHA+18
BB EQU [BP+8]
- = 伪操作（允许重复定义）

.....

EMP = 7

.....

EMP = EMP+1

.....

mov ax,beta+emp

等同于

mov ax,0023H

伪操作

🖥️ 处理器选择伪操作

🖥️ 段定义伪操作

🖥️ 程序开始和结束伪操作

🖥️ 数据定义及存储器分配伪操作

🖥️ 表达式赋值伪操作

🖥️ 地址计数器与对准伪操作

🖥️ 基数控制伪操作

- ❑ ORG伪操作：设置当前地址计数器的值
- ❑ 地址计数器\$：保存当前正在汇编的指令的地址
- ❑ ALIGN伪操作：保证数组边界从2的整数次幂地址开始
- ❑ EVEN伪操作：使下一个变量或指令开始于偶数字节地址

SEG1 SEGMENT

ORG 10

VAR1 DW 1234H

ORG 20

VAR2 DW 5678H

ORG \$+8

VAR3 DW 1357H

ALIGN 4

ARRAY db 100 DUP(?)

A DB 'morning'

EVEN

B DW 2 DUP (?)

SEG1 ENDS

伪操作


 处理器选择伪操作

 段定义伪操作

 程序开始和结束伪操作

 数据定义及存储器分配伪操作

 表达式赋值伪操作

 地址计数器与对准伪操作

 基数控制伪操作

；默认使用十进制，基数为十

```
MOV BX, 0FFH
```

```
MOV BX, 178
```

.RADIX 表达式 ；可以修改基数

```
.RADIX 16
```

```
MOV BX, 0FF
```

```
MOV BX, 178D
```