目录

[一、通用伪指令 1](#_Toc199676304)

[%include 1](#_Toc199676305)

[org 1](#_Toc199676306)

[SECTION 1](#_Toc199676307)

[equ 1](#_Toc199676308)

[二、NASM 特有伪指令 2](#_Toc199676309)

[BITS 2](#_Toc199676310)

[%macro … %endmacro 3](#_Toc199676311)

# 一、通用伪指令

### %include

类似C语言的#include指令

示例：%include "a.inc"

### org

用于设定程序或数据段的起始地址，告诉汇编器接下来的代码或数据应该放置在内存中的哪个位置。

示例：org 07c00h

### SECTION

[SECTION .段名] 用于定义程序中的不同段（section）。它告诉汇编器将接下来的代码或数据放在可执行文件的哪一个逻辑段中

### equ

用于定义常量，基本语法如下：

**符号名 equ 常量表达式**

符号名：可以是一个标识符

常量表达式：可以是一个数字、地址标号之间的差值、算术运算表达式等

# 二、NASM 特有伪指令

### BITS

用于告诉汇编器当前代码段使用的指令集宽度，即生成的是 16 位、32 位还是 64 位的机器码

* 基本语法

[BITS 16]

[BITS 32]

[BITS 64]

这些指令通常放在某个段的开始位置。

* 作用详解

**控制指令解码方式**

不同的 BITS 设置决定了：

* + 使用哪种寄存器（如 AX vs EAX vs RAX）
  + 使用哪种寻址模式（16 位 vs 32 位 vs 64 位地址）
  + 是否使用默认的地址/操作数大小前缀（如 0x66, 0x67）

例如：

mov ax, bx ; [BITS 16] 下是 1 字节长度的指令

; [BITS 32] 下会加上 0x66 前缀表示操作数大小为 16 位

* 使用场景

场景 使用的 BITS 说明

实模式引导代码 [BITS 16] BIOS 启动时运行在实模式下，使用 16 位指令

保护模式内核 [BITS 32] 切换到保护模式后使用 32 位指令

长模式（64位系统） [BITS 64] 现代操作系统进入长模式后使用 64 位指令

### %macro … %endmacro

用于**定义宏**的伪指令。它允许你**将一段常用的汇编代码或数据结构封装成一个可复用的模板**，在多个地方通过调用这个宏来生成对应的代码

* 基本语法

%macro 宏名 参数个数

; 宏体：可以是任意合法的汇编代码或数据定义，会在宏展开时替换其中的内容

%endmacro

示例：定义一个简单的宏

%macro PrintChar 1

mov al, %1 ; 将第一个参数传给 AL

mov ah, 0x0E ; BIOS 的 TTY 输出功能号

int 0x10 ; 调用 BIOS 中断

%endmacro

使用方式：

PrintChar 'A' ; 相当于：

; mov al, 'A'

; mov ah, 0x0E

; int 0x10

* 宏参数引用

在宏体内使用 %1, %2, ... 来引用传递的参数，最多支持 9 个参数。