语法

下面的代码包含汇编语言的基本语法：

```
; 将代码段寄存器和我们的代码段关联起来
assume cs:code

; 代码段开始
code segment
    mov ax, 1122h
    mov bx, 3344h
    add ax, bx

    ; 正常推出程序 相当于 return 0
    mov ah, 4ch
    int 21h

; 代码段结束
code ends

; 程序的结束
end
```

伪指令：如`assume`、`segment`、`ends`、`end`等，没有对应的机器指令，由编译器解析，最终不被CPU执行

segment和ends

`segment`和`ends`的作用是定义一个段，`segment`代表段的开始，`ends`代表段的结束：

```
seg_name segment
; ...
seg_name ends
```

一个有意义的汇编程序中，至少要有一个段做为代码段存放代码。

assume

assume 的作用是将代码段和**mycode**段和CPU中的CS寄存器关联起来。

```
assume cs:mycode
```

end

`end` 代码程序的结束，编译器遇到 `end` 就会结束编译。

退出

下面的代码代表退出程序，使用`int 21`中断

```
; 只要ah是4ch就可以结束
; al是返回码，类似于return 0的0，mov ax, 4c00h
mov ah, 4ch
int 21h
```

定义数据

汇编语言中可以使用`db`、`dw`定义数据：

```
; 定义一个字节的00H
db 0h
; 定义一个字的数据0000H
dw 0h
```

- 在数据段定义数据相当于创建全局变量
- 在栈段定义数据相当于指定栈的容量
- 在代码段定义数据一般不会这样使用

可以使用dup批量的去声明数据：

```
; 声明3个1234H
dw 3 dup(1234H)
```

例子

创建一个包含完整的数据段、代码段、栈段的汇编程序：

```
assume cs: code, ds: data, ss: stack

stack segment
    ; 自定义栈段容量
    db 100 dup(0)
stack ends

data segment
    db 100 dup(0)
data ends

code segment
start:
    mov ax, stack
    mov ss, ax
```

```
    mov ax, data
    mov ds, ax

    mov ax, 1122h
    push ax
    pop bx

    ; 退出
    mov ax, 4c00h
    int 21h
code ends

end start
```