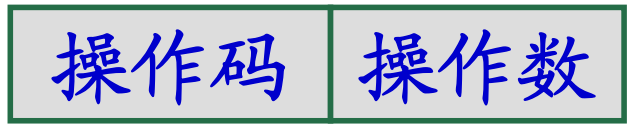
处理器指令格式

处理器指令

- 程序用程序设计语言编写，由指令构成
- 指令由操作码和操作数（地址码）组成
- 操作码（**Opcode**）表明处理器执行的操作
  - 例如数据传送、加法运算、跳转等操作
  - 汇编语言使用指令助记符表示
- 操作数（**Operand**）是参与操作的数据对象
  - 主要以寄存器名或地址形式指明数据的来源
  - 汇编语言使用寄存器、常量、变量等形式表示



使用最多、最基本的数据传送指令

- 传送指令的助记符：**MOV**（取自**Move**）
  - 将数据从一个位置传送到另一个位置
  - 类似高级语言的赋值语句

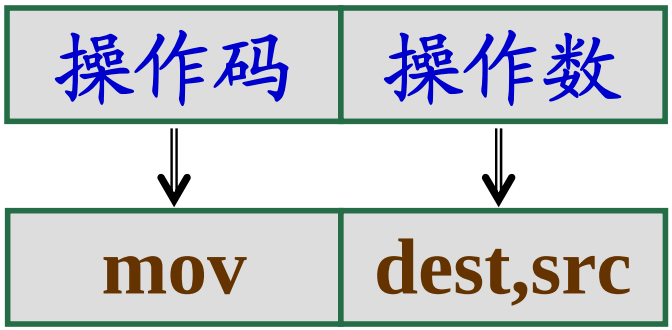
    mov dest,src

  ;源操作数**src**：

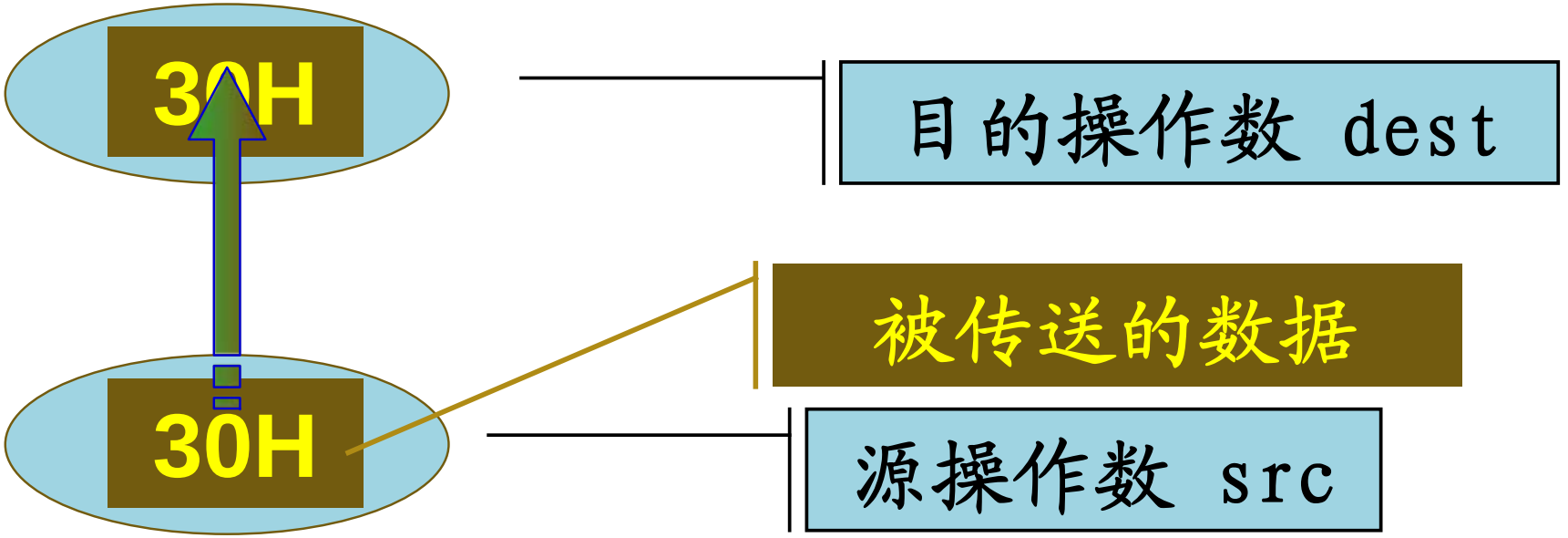
  被传送的数据或数据所在的位置

  ;目的操作数**dest**：

  数据将要传送到的位置



传送指令MOV的功能演示



使用最多、最基本的数据传送指令

- 传送指令的助记符：**MOV**（取自**Move**）
  - 将数据从一个位置传送到另一个位置
  - 类似高级语言的赋值语句

    mov dest,src

  ;源操作数**src**：

  被传送的数据或数据所在的位置

  ;目的操作数**dest**：

  数据将要传送到的位置

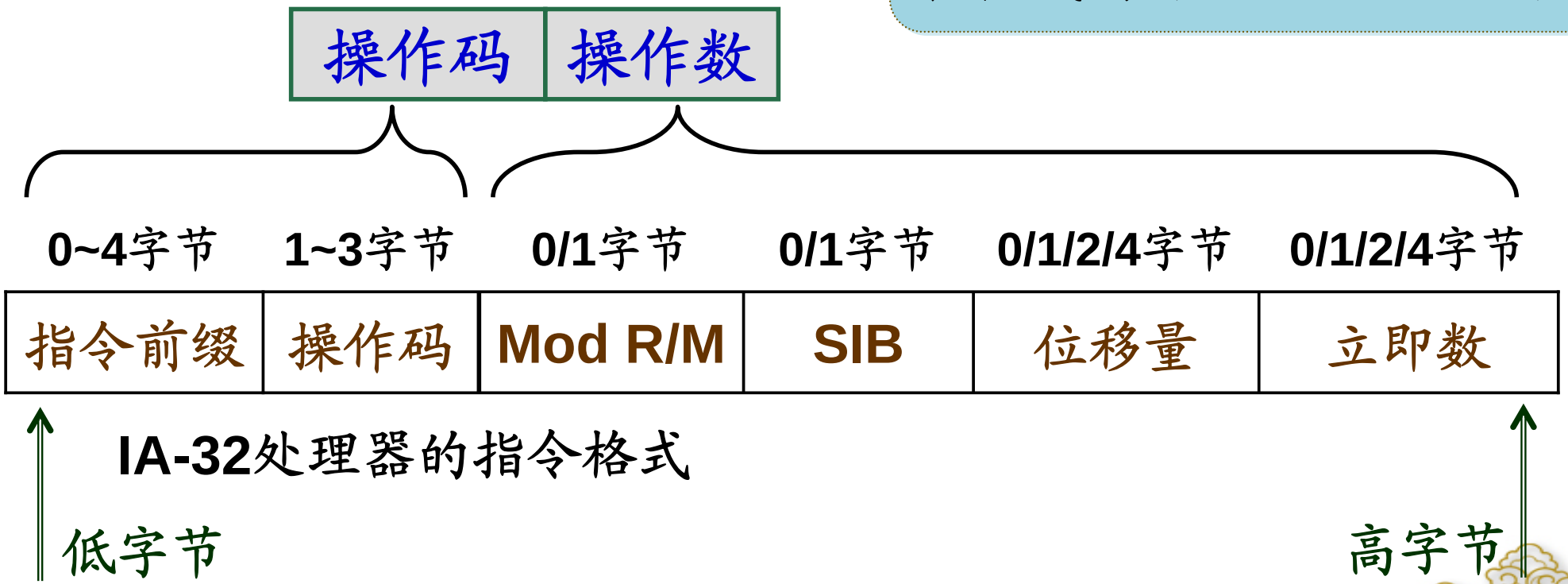
```
mov eax, 100
  ;EAX←100 (常量)
mov eax, dvar
  ;EAX←dvar (变量)
mov eax,ebx
  ;EAX←EBX(寄存器)
```

指令格式（Instruction Format）

- 处理器指令的二进制编码

代码格式（Code Format）
机器代码（Machine Code）

指令的一般格式



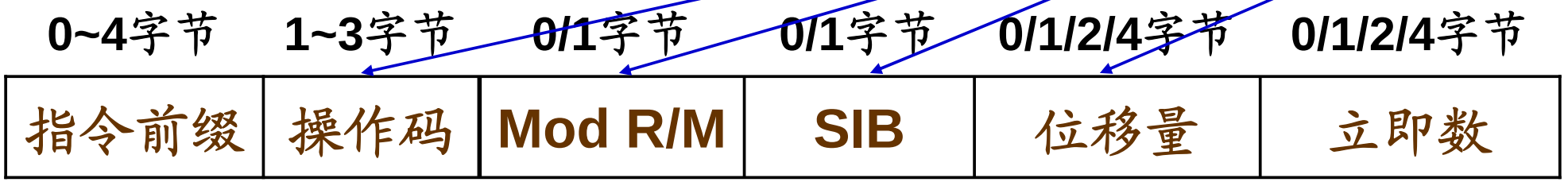
IA-32处理器的指令格式

IA-32处理器指令的代码格式

- IA-32处理器采用可变长度指令格式
- 操作码
  - 可选的指令前缀（用于扩展指令功能，0～4字节）
  - 1～3字节的主要操作码
- 操作数
  - 可选的寻址方式域（包括ModR/M和SIB字段，0或1字节）
  - 可选的位移量（0、1、2或4字节）
  - 可选的立即数（0、1、2或4字节）

IA-32处理器指令格式

```
mov eax,ebx                 ;机器代码：8B C3
mov eax,[ebx]               ;机器代码：8B 03
mov eax,[ebx+esi*4+80h]     ;机器代码：8B 84 B3 80 00 00 00
```



IA-32处理器的指令格式

低字节

高字节

汇编语言语句格式

汇编语言语句

- 源程序由语句组成
- 通常一个语句常占一行（支持续行符“\”）
- 一个语句不超过132个字符，4个部分

- **执行性语句：** 表达处理器指令，实现功能

  标号:　硬指令助记符　操作数, 操作数　;注释

- **说明性语句：** 表达伪指令，控制汇编方式

  名字　　伪指令助记符　参数, 参数, …　　;注释

1. 标号与名字

- 标号：执行性语句中
  - 冒号分隔
  - 表示处理器指令在主存中的逻辑地址
  - 指示分支、循环等程序的目的地址

标号与名字是用户定义的标识符

- 名字：说明性语句中
  - 空格或制表符分隔
  - 变量名、段名、子程序名等
  - 反映变量、段和子程序等的逻辑地址

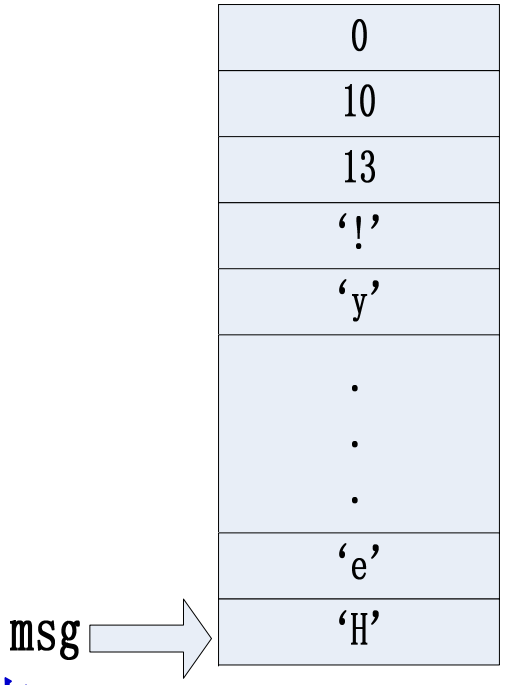
标识符（Identifier）

- 最多由31个字母、数字及规定的特殊符号组成
  - 不能以数字开头
  - 一个源程序中，用户定义的每个标识符必须唯一
  - 不能是保留字(Reserved Word)＝关键字(Key Word)
    - 硬指令助记符： **MOV …**
    - 伪指令助记符： **BYTE …**
    - 操作符： **OFFSET …**
    - 寄存器名： **EAX …**

取名原则类似高级语言，
但默认不区别大小写字母

2. 助记符

- 助记符是帮助记忆指令功能的符号
  - 硬指令助记符表示处理器指令
  - 伪指令助记符表达一个汇编命令
- 处理器指令示例：传送指令 **MOV**
- 伪指令示例：字节变量定义
  - 助记符： **BYTE （或 DB ）**
  - 功能：在主存占用若干存储空间，用于保存变量值




```
msg byte 'Hello, Assembly !',13,10,0
```

3. 操作数和参数

- 处理器指令的操作数：表示参与操作的对象
  - 具体的常量
  - 保存在寄存器的数据
  - 保存在存储器中的变量
  - 逗号前常是目的操作数，逗号后常是源操作数

```
mov eax,offset msg
```

- 伪指令的参数：
  - 常量、变量名、表达式等
  - 可以有多个，参数之间用逗号分隔

```
msg    byte 'Hello, Assembly !',13,10,0
```

4. 注释

- 语句中分号后的内容是注释
  - 对指令或程序进行说明，使用英文或中文均可
  - 汇编程序不对它们做任何处理
  - 注释利于阅读，应养成书写注释的好习惯
  - 注释可以用分号开头，占用一个语句行

```
        ;数据段的变量
msg   byte 'Hello, Assembly !',13,10,0    ;定义字符串
        ;代码段的指令
        mov eax,offset msg    ;EAX获得msg的偏移地址
```

4. 分隔符

- 语句的4个组成部分要用分隔符分开
  - 标号后的冒号
  - 注释前的分号
  - 操作数间和参数间的逗号
  - 分隔其他部分采用一个或多个空格或制表符

分隔符都是英文标点

```
标号: 硬指令助记符  操作数, 操作数    ;注释
名字  伪指令助记符  参数, 参数, ...    ;注释
```

良好的语句格式有利于编程

- 汇编语言不直接支持结构化程序设计
- 为了清晰表达语句，以及整个源程序，建议：
  - 标号和名字从首列开始书写
  - 通过制表符对齐指令助记符和注释部分
  - 助记符与操作数和参数之间用空格或者制表符分隔

标号: 名字	硬指令助记符 操作数, 操作数 伪指令助记符 参数, 参数, …	;注释 ;注释
首列	对齐	对齐

汇编语言语句

;执行性语句

标号:　硬指令助记符　操作数, 操作数　;注释

start:　mov eax,offset msg　;EAX获得msg的偏移地址

;说明性语句

名字　伪指令助记符　参数, 参数, ...　;注释

msg　byte 'Hello, Assembly !',13,10,0　;定义字符串

源程序框架

源程序框架

- 汇编程序为汇编语言制定了严格的语法规范
  - 例如，语句格式、标识符定义、保留字、注释符等
- 汇编程序也为源程序书写设计了框架结构
  - 数据段、代码段等的定义
  - 程序起始执行的位置
  - 汇编结束的标示
  - …

本书的**MASM**源程序框架

基于MASM 6.x简化段定义格式
配合IO32.INC和IO32.LIB文件
具备键盘输入和显示器输出子程序

MASM源程序框架

```
;eg0000.asm in Windows Console
        include io32.inc    ;包含32位输入输出文件
        .data               ;定义数据段
        …                   ;数据定义（数据待填）
        .code               ;定义代码段
start:                      ;程序执行起始位置
        …                   ;主程序（指令待填）
        exit 0              ;程序正常执行终止
        …                   ;子程序（指令待填）
        end start           ;汇编结束
```

包含伪指令INCLUDE

- 用于声明常用的常量定义、过程说明、共享的子程序库等
  - 相当于C和C++语言中，包含头文件的作用

本书的IO32.INC包含文件的前3个语句 **include io32.inc**

```
.686     ;32位指令
.model flat,stdcall
         ;选择平展模型，标准调用规范
option casemap:none
         ;告知MASM区分用户定义标识符的大小写
```

段的简化定义

- MASM支持段的简化定义
  - 数据段定义伪指令

    .DATA ;创建一个数据段
  - 代码段定义伪指令

    .CODE ;创建一个代码段
  - 堆栈段定义伪指令

    .STACK ;创建一个堆栈段

（Windows自动维护堆栈段，用户可以不必设置）

```
include io32.inc
.data
…       ;数据定义
.code
…       ;程序指令
```

程序的开始和结束

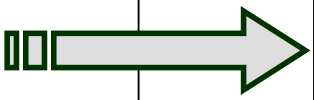
- 程序开始执行的位置
  - 使用一个标号（例如：START）
  - 作为汇编结束END伪指令的参数
- 应用程序执行终止
  - 语句“EXIT 0”终止程序执行
  - 返回操作系统，并提供一个返回代码（0）
- 源程序汇编结束
  - 使用END伪指令语句

```
        .code
start:  …
        exit 0
        …
        end start
```

执行终止≠汇编结束

模板文件eg0000.asm

```
;eg0000.asm in Windows Console
        include io32.inc
        .data
                                ;数据定义
        .code
start:
                                ;主程序
        exit 0
                                ;子程序
        end start
```



本书的简化表达

```
        ;数据段
        …
        ;代码段，主程序
        …
        ;代码段，子程序
        …
```

信息显示程序

第一个程序（C语言）

显示信息

```
Hello, world !
```

```
printf("Hello, world !\n");
```

```
#include <stdio.h>
int main()
{
  printf("Hello, world !\n");
  exit(0);
}
```

信息显示（汇编语言）

- 在数据段给出这个字符串形式的信息：

```
;数据段
msg  byte 'Hello, Assembly!',13,10,0
;定义要显示的字符串
```

"Hello, world !\n"

"\n"

字符串结尾字符

- 在代码段编写显示字符串的程序：

```
;代码段
mov eax,offset msg
call dispmsg
```

printf( );

;指定字符串的偏移地址

;调用I/O子程序显示信息

模板文件eg0000.asm

本书的简化表达

```
;数据段
…
;代码段，主程序
…
;代码段，子程序
…
```

```
;eg0000.asm in Windows Console
        include io32.inc
        .data
                                ;数据定义
        .code
start:
                                ;主程序
        exit 0
                                ;子程序
        end start
```

信息显示的汇编语言源程序

```
;eg0101.asm
        include io32.inc
        .data           ;数据段
msg     byte 'Hello, Assembly!',13,10,0
        .code           ;代码段
start:                  ;程序执行起始位置
        mov eax,offset msg
        call dispmsg
        exit 0          ;程序正常执行终止
        end start       ;汇编结束
```

运行结果

```
Hello, Assembly !
```

输入输出子程序库

- 汇编程序通常不提供任何函数或程序库
- 必须利用操作系统的编程资源
- 本书配套键盘输入和显示器输出的I/O子程序
- 含IO32.INC和IO32.LIB，需要包含文件声明
- 源程序文件开始使用包含命令声明

```
INCLUDE IO32.INC
```

- 子程序调用方法

```
MOV EAX,入口参数
CALL 子程序名
```

字符串显示子程序

子程序名	DISPMSG
入口参数	EAX = 字符串地址
功能说明	显示字符串（以0结尾）

```
        ;数据段，字符串定义
msg  byte 'Hello, Assembly!',13,10,0    ;字符串
        ;代码段，字符串显示
        mov eax,offset msg      ;指定字符串的偏移地址
        call dispmsg            ;调用I/O子程序显示信息
```

常用输出子程序

C语言格式符	子程序名	功能说明
printf("%s",a)	DISPMSG	显示字符串（以0结尾）
printf("%c",a)	DISPC	显示一个字符
printf("\n")	DISPCRLF	光标回车换行，到下行首列
	DISPRD	显示8个32位通用寄存器内容
	DISPRF	显示6个状态标志的状态
printf("%X",a)	DISPHD	以十六进制形式显示8位数据
printf("%u",a)	DISPUID	显示无符号十进制整数
printf("%d",a)	DISPSID	显示有符号十进制整数

常用输入子程序

C语言格式符	子程序名	功能说明
scanf("%s",&a)	READMSG	输入一个字符串（回车结束）
scanf("%c",&a)	READC	输入一个字符（回显）
scanf("%X",&a)	READHD	输入8位十六进制数据
scanf("%u",&a)	READUID	输入无符号十进制整数（$\leqslant 2^{32}-1$）
scanf("%d",&a)	READSID	输入有符号十进制整数（$-2^{31}\sim 2^{31}-1$）

信息显示程序

汇编语言程序

```
        include io32.inc
        .data
msg     byte 'Hello, Assembly!',13,10,0
        .code
start:  mov eax,offset msg
        call dispmsg
        exit 0
        end start
```

C语言程序

```
#include <stdio.h>
int main()
{
  printf("Hello, world !\n");
  exit(0);
}
```