汇编语言与接口技术

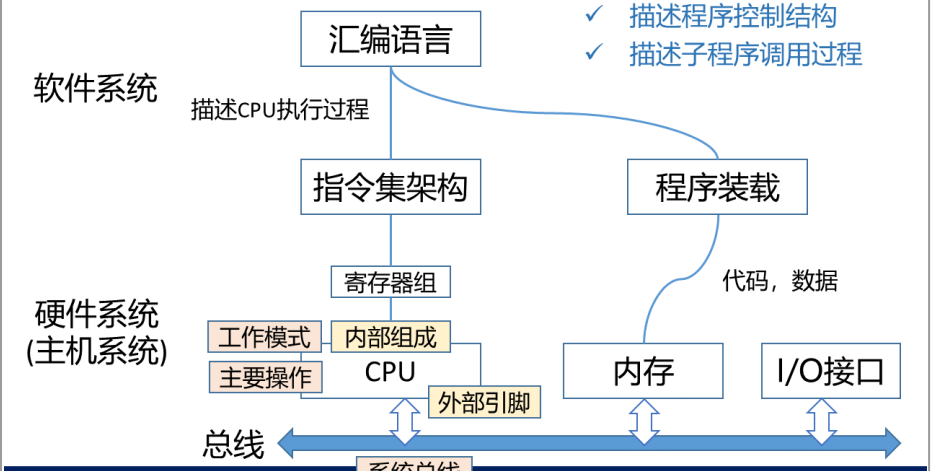
杨春
Chun Yang, Ph.D
chunyang@ustb.edu.cn
北京科技大学计算机科学与技术系
2022-10

第四章 微处理器软件原理 (汇编语言)

学习目的

第四章

- ✓ 描述程序装载过程
- ✓ 描述系统调用过程
- ✓ 描述程序控制结构
- ✓ 描述子程序调用过程



4.2 MASM 语句结构 (描述程序装载过程)

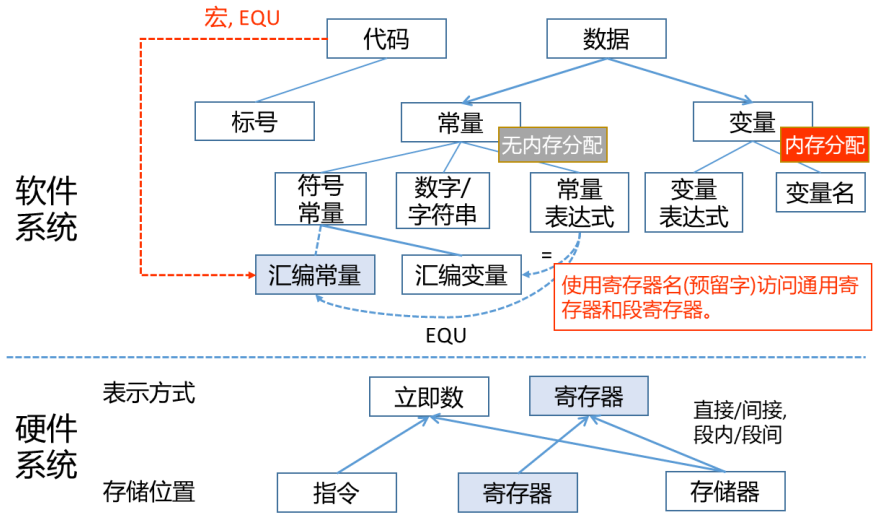
4.2.1 MASM代码文件一般格式

汇编代码文件（*.asm）

```
.model SMALL
stack   SEGMENT
        DB   100  DUP(?)
stack   ENDS
data    SEGMENT
        <数据、变量在此定义>
data    ENDS
code    SEGMENT
        ASSUME  CS:code, DS:data,
ES:data , SS:stack
start:   MOV    AX, data
        MOV    DS, AX
        MOV    ES, AX
        <你的代码在此>
        MOV    AH, 4CH
        INT    21H
code    ENDS
        END    start
```

- ✓ 程序按段编写，与8086内存按段管理对应；
- ✓ 一个源程序由若干段组成，如数据段、代码段和堆栈段等；至少有一个代码段，为确保程序的执行和正确返回，需要堆栈段；
- ✓ 每段由数条语句构成，每条语句占一行，指令性语句和指示性语句(伪指令语句)；
- ✓ 程序中设有返回DOS的功能，使程序执行完后返回DOS操作系统；
- ✓ 程序最后为 END 结束语句，后跟程序启动地址，启动地址指示程序开始执行的第一条语句；
- ✓ 对相应的段寄存器赋值，程序中用到内存操作数时，应按操作数的寻址方式对段寄存器赋值。

常量 v.s. 变量



4.2.4 存储模式伪指令/段定义伪指令

屏幕上显示字符串1357??????????　例 4.3

- 计算机中字符、数码转换的处理
  - 计算机处理字符时，常用的字符编码是ASCII码，数字和字母的ASCII码是一个有序序列

    数字0~9 ： 30H ~ 39H
    大写字母A~Z ： 41H ~ 5AH
    小写字母a~z ： 61H ~ 7AH
  - 计算机处理信息时，其对象都是二进制数，外设(显示器、打印机、键盘等) 用ASCII码与CPU进行信息传送。
    - 键盘上按下某一字符键（如’9’），键盘接口向键盘缓冲区送去的是该字符的ASCII码(如39H)，不是送数字09H。
    - 文本方式下，要在显示器上显示某一字符(如’A’)，须将该字符的ASCII码(如41H)送显示缓冲区，不是送数字0AH。

4.2.4 存储模式伪指令/段定义伪指令

屏幕上显示字符串1357??????????　例 4.3

需将该字符串每个字符的ASCII码值存入存储单元，然后调用显示字符串的中断，在显示器上显示字符串，其代码如下所示：

```
.MODEL  SMALL
.STACK
.DATA
    V_BYTE     EQU     THIS BYTE
    V_WORD     DW      3332H, 3735H
    TARGET     DW      5 DUP (20H)
    GRLF       DB      0DH, 0AH, '$'
    FLAG       DB      0
    N_POINT    DW      OFFSET S_LABEL
```

	存储单元	偏移地址
N_POINT	（地址）	0012H
FLAG	00H	0011H
	'$'	0010H
	0AH	000FH
GRLF	0DH	000EH
	00H	000DH
	20H	000CH
	00H	000BH
	20H	000AH
	00H	0009H
	20H	0008H
	00H	0007H
	20H	0006H
	00H	0005H
TARGET	20H	0004H
	37H	0003H
	35H	0002H
	33H	0001H
V_BYTE V_WORD	32H	0000H

4.2.4 存储模式伪指令/段定义伪指令

屏幕上显示字符串1357??????????　例 4.3

```
.CODE
.STARTUP
        MOV     AL, BYTE PTR V_WORD
        DEC     AL
        MOV     V_BYTE, AL
N_LABEL: CMP    FLAG, 1
        JZ      S_LABEL
        INC     FLAG
        JMP     SHORT N_LABEL
S_LABEL: CMP    FLAG, 2
        JZ      NEXT
        INC     FLAG
        JMP     N_POINT                 ; S_LABEL
```

	存储单元	偏移地址
N_POINT	（地址）	0012H
FLAG	00H	0011H
	'$'	0010H
	0AH	000FH
GRLF	0DH	000EH
	00H	000DH
	20H	000CH
	00H	000BH
	20H	000AH
	00H	0009H
	20H	0008H
	00H	0007H
	20H	0006H
	00H	0005H
TARGET	20H	0004H
	37H	0003H
	35H	0002H
	33H	0001H
V_BYTE V_WORD	31H	0000H

4.2.4 存储模式伪指令/段定义伪指令

屏幕上显示字符串1357??????????　例 4.3

```
; .CODE
NEXT:   MOV  AX, TYPE  V_WORD       ; MOV AX,2
        MOV  CX, LENGTH  TARGET     ; MOV CX,5
        MOV  SI, OFFSET  TARTGET
W_AGAIN: MOV  [SI], AX              ; 对字进行操作
        INC   SI                    ; SI 指针+2
        INC   SI
        LOOP  W_AGAIN
```

	存储单元	偏移地址
N_POINT	（地址）	0012H
FLAG	00H	0011H
	'$'	0010H
	0AH	000FH
GRLF	0DH	000EH
	00H	000DH
	02H	000CH
	00H	000BH
	02H	000AH
	00H	0009H
	02H	0008H
	00H	0007H
	02H	0006H
	00H	0005H
TARGET	02H	0004H
	37H	0003H
	35H	0002H
	33H	0001H
V_BYTE V_WORD	31H	0000H

4.2.4 存储模式伪指令/段定义伪指令

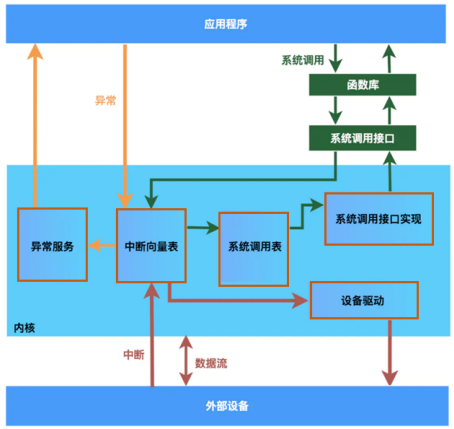
屏幕上显示字符串1357??????????　例 4.3

```
; .CODE
        MOV     CX, SIZE  TARGET
        MOV     AL, '?'
        MOV     DI, OFFSET  TARGET
B_AGAIN: MOV    [DI], AL            ; 对字节进行操作
        INC     DI                  ; DI 指针+1
        LOOP    B_AGAIN
        MOV     DX, OFFSET V_WORD
        MOV     AH, 9
        INT     21H
END
```

	存储单元	偏移地址
N_POINT	（地址）	0012H
FLAG	00H	0011H
	'$'	0010H
	0AH	000FH
GRLF	0DH	000EH
	'?'	000DH
	'?'	000CH
	'?'	000BH
	'?'	000AH
	'?'	0009H
	'?'	0008H
	'?'	0007H
	'?'	0006H
	'?'	0005H
TARGET	'?'	0004H
	37H	0003H
	35H	0002H
	33H	0001H
V_BYTE V_WORD	31H	0000H

4.3 系统调用
(描述系统调用过程)

4.3.1 系统调用 (中断机制)

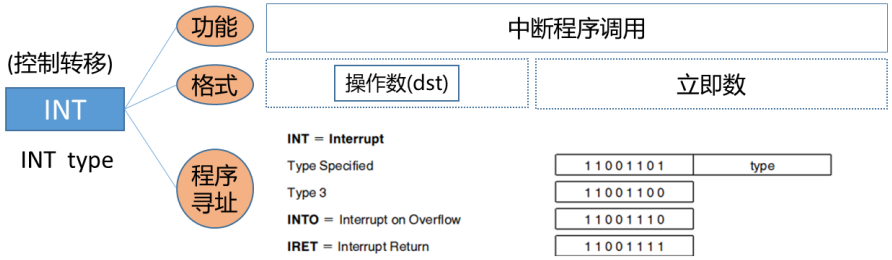


- 操作系统是中断驱动的。
- 中断机制指cpu在执行某件事情时可以暂停执行当前的任务而转去执行其他任务的能力。
- 系统调用(软件中断,绿色箭头):
  ① 采用(中断向量表)相同入口进入系统调用
  ② 在系统调用表上根据参数不同进行具体的调用服务选择。
  ③ 系统调用的中断与硬件无关，属于自愿性中断事件，应用程序主动向操作发出的服务请求，语义为请求os的服务。

https://www.jianshu.com/p/643cac5f5f02

控制转移指令——中断控制INT

- 中断调用指令INT,提供中断调用程序类型type。
- INT 21H是DOS系统调用，INT 05H~1FH 是BIOS系统调用。
- IRET对应中断程序返回。



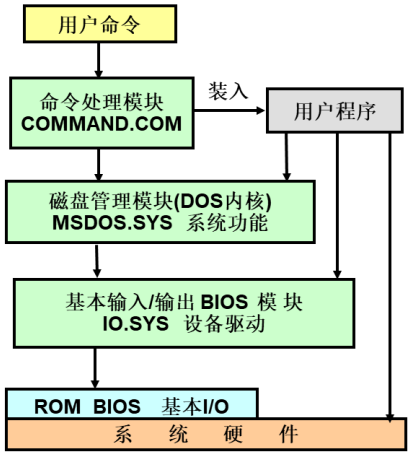
4.3.1 DOS系统调用

DOS系统调用概述

DOS系统调用提供许多子程序供用户调用，包括：

- 磁盘管理
- 内存管理
- 基本输入输出管理
- ...

使用中断指令 INT 21H完成相应子程序的调用，书表4.5系统功能调用说明。



4.3.1 DOS系统调用

- 汇编程序的运行结果，或是保留在寄存器中，或是保留在存储器中，不能方便直观的看到
- 使用INT 21H完成子程序调用需要入口参数及出口参数
- 基本I/O功能调用见书表4.6

功能号	功能描述	使用说明
01H	键盘输入，屏幕显示	入口参数：无 出口参数：AL存放输入字符
03H	异步通信输入	入口参数：无 出口参数：AL存放异步通信口接收的数据
04H	异步通信输出	入口参数：DL存放要输出的数据 出口参数：无
05H	打印输出	入口参数：DL存放待打印的字符 出口参数：无
09H	输出字符串	入口参数：DS:DX指向内存中一个以$结束的字符串 出口参数：无
0AH	键盘接收字符串，存内存缓冲区	入口参数：DS:DX指向输出缓冲区 出口参数：DS:DX指向输出缓冲区
2AH	读取日期	入口参数：无 出口参数：CX：年；DH：月；DL：日
2BH	设置日期	入口参数：CX：年；DH：月；DL：日 出口参数：AH=00，设置成功；AH=0FFH，无效

4.3.1 DOS系统调用

INT 21H

通常按照如下4个步骤进行：

(1) 在AH寄存器中设置系统功能调用号

(2) 在指定寄存器中设置入口参数

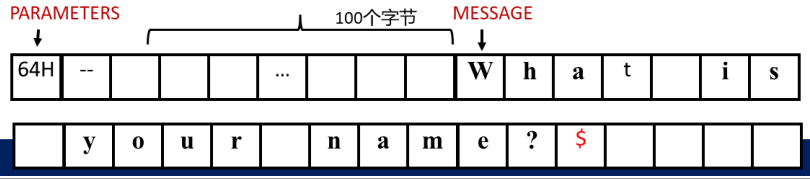
(3) 执行指令INT 21H实现中断服务程序的功能调用

(4) 根据出口参数分析功能调用执行情况

4.3.1 DOS系统调用

例：利用DOS调用在屏幕上显示提示信息，然后接收键盘输入信息，并存入缓冲区。

```
PARAMETERS    DB    100                          ; 数据定义
              DB    ?
              DB    100 DUP(?)
MESSAGE       DB    'What is your name?'
              DB    '$'
DISP:   MOV  DX, OFFSET MESSAGE                  ; 屏幕显示
        MOV  AH, 09H
        INT  21H
KEY:    MOV  DX, OFFSET PARAMETERS               ; 键盘输入
        MOV  AH, 0AH
        INT  21H
```

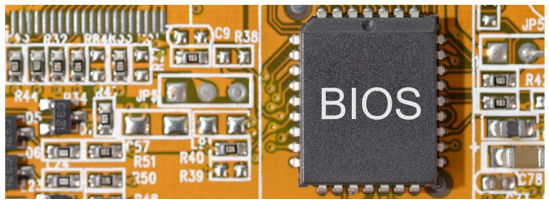


4.3.1 DOS系统调用

例　利用键盘输入的字符产生分支：

```
        MOV   AH, 1        ;等待从键盘输入
        INT    21H
        CMP    AL, 'Y'     ;是' Y'?
        JZ     yes
        CMP    AL, 'y'     ;是' y'?
        JZ     yes
no:     . . .
        . . .
        JMP    exit
yes:    . . .
        . . .
exit:   . . .
```

4.3.2 BIOS中断调用



BIOS (Basic I/O System)

是固化在ROM中的一组I/O设备驱动程序，为系统各主要部件提供设备级的控制，负责管理系统内的输入输出设备，直接为DOS操作系统和应用程序提供底层设备驱动服务。

UEFI (Unified Extensible Firmware Interface Specification)

4.3.2 BIOS中断调用

大多数以软件中断方式调用，少数以硬件中断调用。

调用格式：

INT n ；n=05H~1FH

常用BIOS服务功能见书P144，表4.7

BIOS服务	功能号	功 能
打印屏幕服务	05H	当前视频内容送默认打印机
视频服务	10H	为显示适配器提供I/O支持
硬盘服务	13H	提供硬盘的读、写、格式化、初始化、诊断
串行通信服务	14H	为串行适配器提供字符输入输出
键盘服务	16H	为键盘提供I/O支持
并行打印机服务	17H	为并行打印机提供I/O支持
日期时间服务	1AH	设置和读取时间、日期、声源等(可用于随机数生成)

4.3.2 BIOS中断调用

- 视频服务

视频服务由INT 10H 启动，一般的步骤是：

① AH选择视频服务功能，AL或BL选择子功能
② AL存放待显示字符或像素值
③ 功能调用时，保存BX, CX, DX及段寄存器值
④ X坐标在CX存放（图形显示）；在DL存放 （正文显示）
⑤ BH存放显示页（0开始计数）

例：光标移到3行14列

```
MOV     AH, 02H
MOV     DH, 3
MOV     DL, 14
INT     10H
```

4.3.2 BIOS中断调用

- 键盘服务

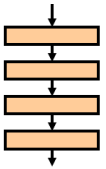
例：16H键盘中断处理
操作系统和应用程序调用INT 16H指令，对键盘进行检测和设置，并读取键盘缓冲区中的键码，有4种功能。

功能号	功 能	返回参数
AH=0	键盘缓冲区读取1个字符的键码送到AX	AH=系统扫描码 AL=字符的ASCII或0
AH=1	检测键盘缓冲区中是否有键码	ZF=0则有键码并读入AX ZF=1则无键码
AH=2	读取特殊键的状态标志	AL中为读取的状态标志
AH=3	设置键盘速率和延迟时间 BL=速率，BH=延迟时间	无

4.4 MASM程序设计 (描述程序控制结构)

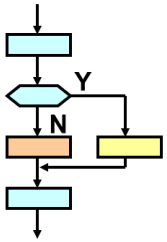
4.4.1 汇编程序结构

顺序



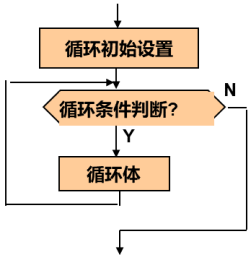
```
MOV  AX，DATA
MOV  DS，AX
MOV  CL，M
MOV  AX，0
MOV  SI，AX
```

分支




```
...
CMP  AL, BL
JG   great
     AL≤BL处理
JMP  exit
great:  AL>BL处理
exit:   ...
```

循环




```
     MOV  CL，M
     MOV  AX，0
     MOV  SI，AX
XUN：ADD  AX，ARRAY[SI]
     ADD  SI，2
     LOOP XUN
```

4.4.2 顺序结构

例：查表的方法将一位十六进制数（0~9，A~F）转换成相应的ASCII码。

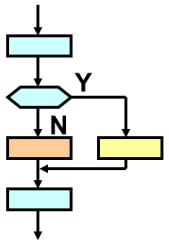
```
DATA    SEGMENT
   TABLE  DB   30H,31H,32H,33H,34H,35H,36H,37H,38H,39H,41H,42H,43H,44H,45H,46H
    HEX   DB   4
    ASC   DB   ?
DATA    ENDS

STACK  SEGMENT
    DW  20H DUP(0)
STACK  ENDS

COSEG   SEGMENT
    ASSUME  CS:COSEG,DS:DATA,SS:STACK
START:  MOV   AX, DATA
        MOV   DS, AX
        LEA   BX, TABLE
        XOR   AH, AH
        MOV   AL, HEX
        ADD   BX, AX
        MOV   AL, [BX]
        MOV   ASC, AL
COSEG   ENDS
        END START
```

4.4.3 分支结构

分支




```
...
CMP  AL, BL
JG   great
     AL≤BL处理
JMP  exit
great:  AL>BL处理
exit:   ...
```

- 分支（选择）结构依控制模块执行分支。
- 依赖指令：
  - ✓ 无条件跳转指令JMP
  - ✓ 有条件跳转指令JCC
- 既可以进行段内跳转，也可以进行段间跳转。

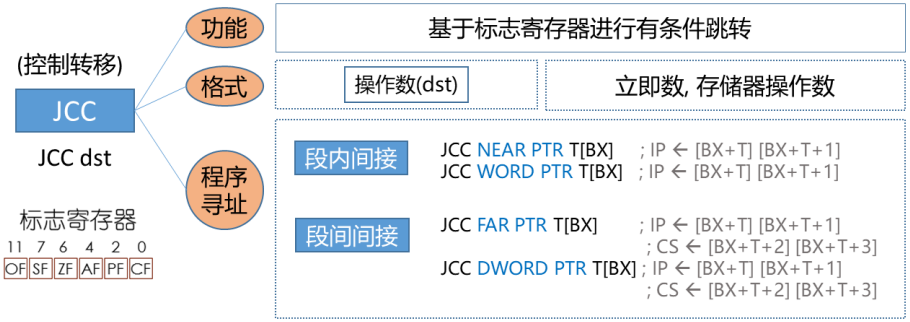
3.6.5 控制转移指令——JMP

- 无条件跳转指令JMP, 跳转至地址 CS:[IP]。
- 段内寻址只修改IP, 段间寻址既修改IP也修改CS。

SHORT	相对于当前指令地址 -128~127
NEAR	段内转移，CS不变，指令给出IP
FAR	段间转移，CS、IP均需给出

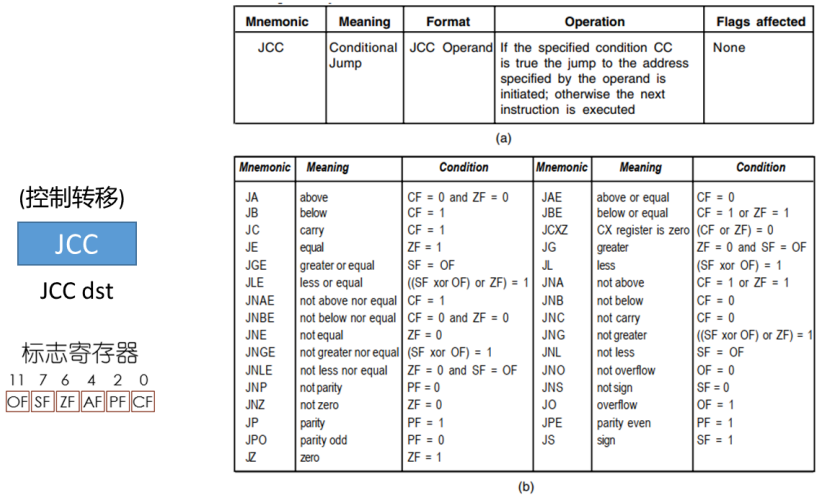
控制转移指令——JCC

- 有条件跳转指令JCC, 基于标志寄存器跳转至地址 CS:[IP]。
- 段内寻址只修改IP, 段间寻址既修改IP也修改CS。



SHORT 相对于当前指令地址 -128~127
NEAR 段内转移，CS不变，指令给出IP
FAR 段间转移，CS、IP均需给出

控制转移指令——JCC



Mnemonic	Meaning	Format	Operation	Flags affected
JCC	Conditional Jump	JCC Operand	If the specified condition CC is true the jump to the address specified by the operand is initiated; otherwise the next instruction is executed	None

(a)

Mnemonic	Meaning	Condition	Mnemonic	Meaning	Condition
JA	above	CF = 0 and ZF = 0	JAE	above or equal	CF = 0
JB	below	CF = 1	JBE	below or equal	CF = 1 or ZF = 1
JC	carry	CF = 1	JCXZ	CX register is zero	(CF or ZF) = 0
JE	equal	ZF = 1	JG	greater	ZF = 0 and SF = OF
JGE	greater or equal	SF = OF	JL	less	(SF xor OF) = 1
JLE	less or equal	((SF xor OF) or ZF) = 1	JNA	not above	CF = 1 or ZF = 1
JNAE	not above nor equal	CF = 1	JNB	not below	CF = 0
JNBE	not below nor equal	CF = 0 and ZF = 0	JNC	not carry	CF = 0
JNE	not equal	ZF = 0	JNG	not greater	((SF xor OF) or ZF) = 1
JNGE	not greater nor equal	(SF xor OF) = 1	JNL	not less	SF = OF
JNLE	not less nor equal	ZF = 0 and SF = OF	JNO	not overflow	OF = 0
JNP	not parity	PF = 0	JNS	not sign	SF = 0
JNZ	not zero	ZF = 0	JO	overflow	OF = 1
JP	parity	PF = 1	JPE	parity even	PF = 1
JPO	parity odd	PF = 0	JS	sign	SF = 1
JZ	zero	ZF = 1			

(b)

Fig. 14.14: (a) Conditional jump instruction (b) Types of conditional jump instructions

4.4.3 分支结构

注意：分支的开始点和结束点

例4.22：计算AX中符号数绝对值

```
        CMP     AX, 0
        JGE     NONEG  ; >=

        NEG     AX
NONEG:  MOV     RESULT, AX
```

```
        CMP     AX, 0
        JL      YESNEG ; <
        JMP     NONEG
YESNEG: NEG     AX
NONEG:  MOV     RESULT, AX
```

例4.23：显示BX最高位

```
        SHL     BX, 1
        JC      ONE    ;CF=1
        MOV     DL, '0'
        JMP     TWO
ONE:    MOV     DL, '1'
TWO:    MOV     AH, 2
        INT     21H
```

```
        MOV     DL, '0'
        SHL     BX, 1
        JNC     TWO    ;CF=0
        MOV     DL, '1'
TWO:    MOV     AH, 2
        INT     21H
```


4.4.3 分支结构

例:

```
GMAX:   MOV     BX, 2000H
        MOV     AL, BYTE PTR [BX]
        MOV     CX, 14H
P1:     CMP     AL, BYTE PTR [BX]
        JAE     P2                  ; >=, Above or equal
        MOV     AL, BYTE PTR [BX]
P2:     INC     BX
        DEC     CX
        JNZ     P1                  ; ZF=0
        MOV     BX, 3000H
        MOV     BYTE PTR [BX], AL
```

该程序段功能?

4.4.3 分支结构

例4.24：判断AX +BX+C=-是否有实根，若有实根，则将字节变量TAG置1，否则置0。假设 A、B、C均为字节变量，数据范围为-128~+127。

```
.MODEL  SMALL
.STACK
.DATA
    _A          DB      ?
    _B          DB      ?
    _C          DB      ?
    TAG         DB      ?
```

```
.CODE
.STARTUP
        MOV     AL, _B
        IMUL    AL
        MOV     BX, AX
        MOV     AL, _A
        IMUL    _C
        MOV     CX, 4
        IMUL    CX
        CMP     BX, AX
        JGE     YES         ;>=
        MOV     TAG, 0
        JMP     DONE
YES:    MOV     TAG, 1
DONE:   .EXIT 0
        END
```


4.4.3 分支结构

例4.25：根据键盘输入的1~8数字转向8个不同的处理程序段。

```
.MODEL  SMALL
.STACK
.DATA
   MSG    DB   'Input number(1~8):', 0DH, 0AH, $
   MSG1   DB   'Chapter1: ...', 0DH, 0AH, $
   MSG2   DB   'Chapter2: ...', 0DH, 0AH, $
   MSG3   DB   'Chapter3: ...', 0DH, 0AH, $
   MSG4   DB   'Chapter4: ...', 0DH, 0AH, $
   MSG5   DB   'Chapter5: ...', 0DH, 0AH, $
   MSG6   DB   'Chapter6: ...', 0DH, 0AH, $
   MSG7   DB   'Chapter7: ...', 0DH, 0AH, $
   MSG8   DB   'Chapter8: ...', 0DH, 0AH, $
TABLE   DW   DISP1,DISP2,DISP3,DISP4,DISP5,DISP6,DISP7,DISP8
```

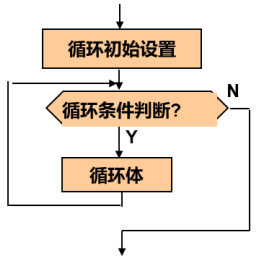

4.4.3 分支结构

例4.25：根据键盘输入的1~8数字转向8个不同的处理程序段。

```
.CODE
.STARTUP
START1:MOV      DX, OFFSET MSG
       MOV      AH, 9
       INT      21H
       MOV      AH, 1
       INT      21H
       CMP      AL, '1'
       JB       START1      ;<
       CMP      AL, '8'
       JA       START1      ;>
       AND      AX, 000FH
       DEC      AX
       SHL      AX, 1
       MOV      BX, AX
       JMP      TABLE[BX]
START2: MOV     AH, 9
        INT     21H
.EXIT 0
```

```
DISP1:  MOV     DX, OFFSET MSG1
        JMP     START2
DISP2:  MOV     DX, OFFSET MSG2
        JMP     START2
DISP3:  MOV     DX, OFFSET MSG3
        JMP     START2
.......
END
```

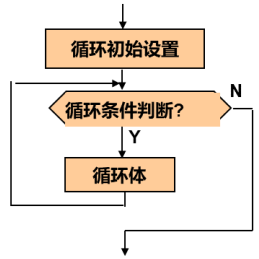

4.4.4 循环结构




```
        MOV  CL, M
        MOV  AX, 0
        MOV  SI, AX
XUN:    ADD  AX, ARRAY[SI]
        ADD  SI, 2
        LOOP XUN
```

- 循环控制结构依照控制模块重复执行循环体。
- 显式循环的控制模块和循环体分离，包括计数控制和条件控制。
- 递归，也就是一个函数直接或间接调用自身。(4.5 子程序调用)。

4.4.4 循环结构

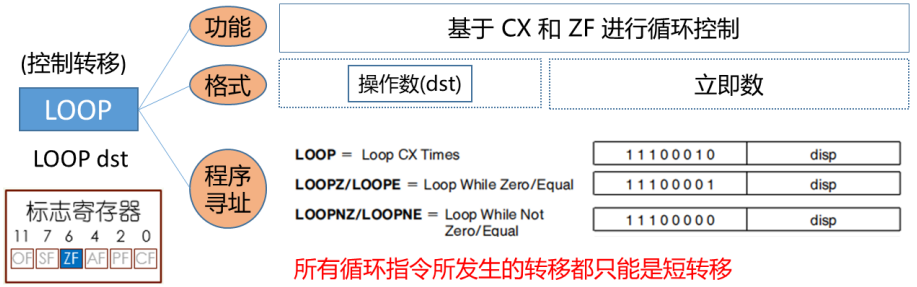



```
      MOV  CL，M
      MOV  AX，0
      MOV  SI，AX
XUN：ADD  AX，ARRAY[SI]
      ADD  SI，2
      LOOP XUN
```

- 循环控制结构依照控制模块重复执行循环体。
- 显式循环的控制模块和循环体分离，包括计数控制和条件控制。
- 依赖指令：
  - ✓ 无条件跳转指令JMP
  - ✓ 有条件跳转指令JCC
  - ✓ 循环控制指令LOOP
- JMP/JCC既可以进行段内跳转，也可以进行段间跳转。
- LOOP所发生的转移都只能是短转移。

控制转移指令——循环

- 循环指令包括无条件循环LOOP，有条件循环LOOPZ/LOOPE和LOOPNZ/LOOPNE。
- 循环终止条件：① CX=0;
  ② ZF=0 (LOOPZ/LOOPE)或ZF=1 (LOOPZ/LOOPE)



所有循环指令所发生的转移都只能是短转移

4.4.4 循环结构

■注意：循环程序的循环条件和结束点

例4.27：计算1~100数字之和，结果存入字变量SUM。

```
.MODEL SMALL
.STACK
.DATA
    SUM         DW      ?
.CODE
.STARTUP
        XOR     AX, AX
        MOV     CX, 100
AGAIN:  ADD     AX, CX
        LOOP    AGAIN
        MOV     SUM, AX
.EXIT 0
END
```

4.4.4 循环结构

例:　5个连续字节单元的累加

```
DATA    SEGMENT             ;定义数据段
   D1     DB   5 DUP (?)
   CC     EQU  $－D1
  SUM     DW   0
DATA    ENDS                ;数据段定义结束
STACK SEGMENT               ;定义堆栈段
  DB  100  DUP (?)
STACK   ENDS                ;堆栈段定义结束
```

```
CODE    SEGMENT             ;定义代码段
ASSUME DS:DATA，SS:STACK，CS:CODE
BEGIN:  MOV     AX，DATA
        MOV     DS，AX
        LEA     SI，D1
        MOV     CX，CC
        CLD             ;(DF)=0，串指针自动增量
        LODS    BYTE PTR [SI]
        MOV     AH，0       ;清AX的高字节
        CLC                 ;清CF位
AGAN:   ADC     SUM，AX
        LODS    BYTE PTR [SI]
        LOOP    AGAN
        MOV     AH，4CH     ;完成，返回DOS
        INT     21H
CODE    ENDS                ;代码段结束
        END     BEGIN       ;整个程序结束
```