



東北大學
Northeastern University

汇编语言程序设计

主讲：东北大学计算机学院 刘松冉

第六章 分支结构程序

一. 分支结构程序的引出

二. 转移指令

三. 分支结构程序设计

四. 多分支结构程序设计



一. 分支结构程序的引出

▶ 用计算机处理问题过程中，总是要求计算机能做出各种逻辑判断，并根据判断的结果，做相应的处理。

例如，火车站用计算机计算托运行李的托运费，当旅客行李重量小于或等于20kg时，收费0.2元/kg，当行李重量超过20kg时，20kg以内部分0.2元/kg，超出部分，收费0.3元/kg。这个处理过程，可归纳为下面数学表达式：

$$\begin{cases} 0.2w(w \leq 20\text{kg}) \\ 0.2 \times 20 + 0.3 \times (w - 20)(w > 20\text{kg}) \end{cases}$$

二. 转移指令

- ▶ 1. 无条件转移指令
- 2. 条件转移指令

二. 转移指令 – 1. 无条件转移指令

▶ JMP

- 指令汇编格式: JMP target
- 操作：1) 段内转移：IP←目标的偏移地址；
2) 段间转移：IP←目标的偏移地址
CS←目标所处代码段的基址
- 受影响的状态标志位：没有
- 说明：指令中的target可以是直接标号、寄存器间接寻址或存储器间接寻址形式。

二. 转移指令 – 1. 无条件转移指令

▶ JMP – 1) 段内转移 (NEAR)

a) 段内直接转移

JMP LABEL_N ; LABEL_N在当前代码段

JMP SHORT LABEL_N ; LABEL_N在当前代码段, 且在-128~127范围内;

二. 转移指令 – 1. 无条件转移指令

▶ JMP – 1) 段内转移 (NEAR)

a) 段内直接转移

➤ 例：

0000	CSEG	SEGMENT
		ASSUME CS:CSEG
0000 E9010AR	START:	JMP L1
0003 EB05		JMP SHORT L2
0005 EB0390		JMP L2
0008 EBF6		JMP START
000A 90	L2:	NOP
010A		ORG L2+100H
010A 90	L1:	NOP
010B	CSEG	ENDS
		END

二. 转移指令 – 1. 无条件转移指令

▶ JMP – 1) 段内转移 (NEAR)

a) 段内直接转移

```
JMP LABEL_N ; LABEL_N在当前代码段  
JMP SHORT LABEL_N ; LABEL_N在当前代码段, 且在-128~127范围内;
```

b) 段内寄存器间接转移

```
JMP AX
```

c) 段内存储器间接转移

```
JMP [SI]  
JMP WORD PTR [BX+DI+1000H]
```


二. 转移指令 – 1. 无条件转移指令

▶ JMP – 2) 段间转移 (FAR)

a) 段间直接转移

JMP LABEL_F ; LABEL_F 是其他代码段的机器指令代码，并已用 PUBLIC 说明，
; 在本段中已用 EXTRN 说明，未说明的不能引用

b) 段间存储器间接转移

JMP DWORD PTR [SI] ; IP ← (SI, SI+1)
; CS ← (SI+2, SI+3)
JMP VAR_DW ; VAR_DW 为双字类型的标号
; IP ← (VAR_DW, VAR_DW+1)
; CS ← (VAR_DW+2, VAR_DW+3)
JMP DWORD PTR [BX][DI] ; IP ← (BX+DI, BX+DI+1)
; CS ← (BX+DI+2, BX+DI+3)

段间间接寻址只能通过存储器来实现。它是将指令中给出的第一个字送到 IP 中，第二个字送入到 CS 中。

二. 转移指令 – 2. 条件转移指令

▶ 条件转移指令是根据CPU中状态标志位的状态决定程序执行的流程，既可能产生程序转移，也可能不产生程序转移。条件转移指令是以对不同的状态标志的测试为条件。

如果条件成立，则控制转移到指令中所给出的转移目标。条件不成立，程序将顺序执行。所有的条件转移指令均为短(short)转移。

二. 转移指令 – 2. 条件转移指令

▶ 1) 根据单标志转移的指令

CF	JC	lab	CF=1
	JNC	lab	CF=0
ZF	JZ	lab	ZF=1
	JNZ	lab	ZF=0
SF	JS	lab	SF=1
	JNS	lab	SF=0
OF	JO	lab	OF=1
	JNO	lab	OF=0
PF	JP/JPE	lab	PF=1
	JNP/JPO	lab	PF=0

Even

Odd

二. 转移指令 – 2. 条件转移指令

▶ 2) 根据两数 (A, B) 的大小关系转移的指令

a) 使用该指令前用过比较 (CMP A,B)、减法 (SUB A,B; SBB A,B) 指令

b) A与B的关系共有6种：

$A < B$ $A \geq B$

$A \leq B$ $A = B$

$A > B$ $A \neq B$

c) 比较转移时分无符号数和带符号数。

二. 转移指令 – 2. 条件转移指令

▶ 2) 根据两数 (A, B) 的大小关系转移的指令

	无符号数		带符号数	
$A < B$	JB	CF=1	JL	$(SF \wedge OF)=1$
$A \leq B$	JBE	$CF=1 \parallel ZF=1$	JLE	$(SF \wedge OF) \parallel ZF=1$
$A > B$	JA	$CF=0 \& ZF=0$	JG	$(SF \wedge OF) \parallel ZF=0$
$A \geq B$	JAE	CF=0	JGE	$(SF \wedge OF)=0$
$A = B$	JE (ZF=1)			
$A \neq B$	JNE (ZF=0)			

Below Above Less Great Equal

三. 分支结构程序设计

▶ 例 6.1 设X为单字节带符号整数,且存于ARGX单元,计算结果Y存入RLT单元。

$$Y = \begin{cases} X+10 & (0 < X \leq 8) \\ 5X-2 & (8 < X < 15) \\ |X| & (\text{其它}) \end{cases}$$

三. 分支结构程序设计

▶ 例 6.1 设X为单字节带符号整数,且存于ARGX单元,计算结果Y存入RLT单元。

$$Y = \begin{cases} X+10 & (0 < X \leq 8) \\ 5X-2 & (8 < X < 15) \\ X & (X=0, X \geq 15) \\ -X & (X < 0) \end{cases}$$

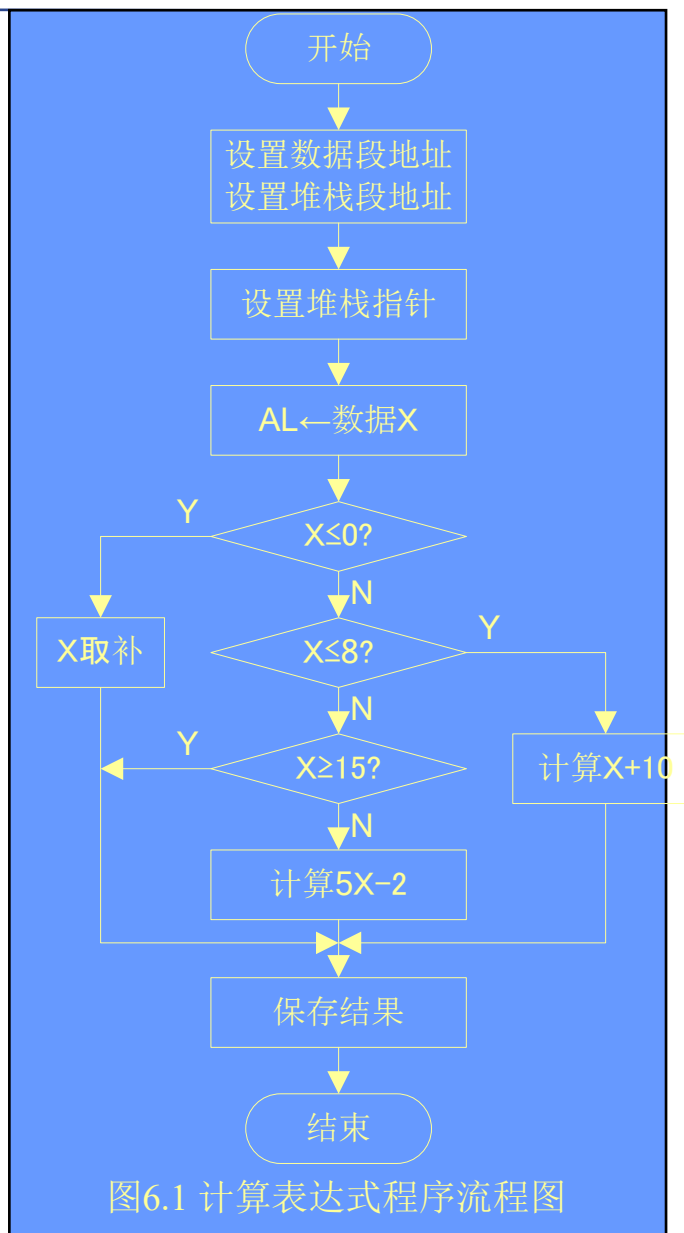
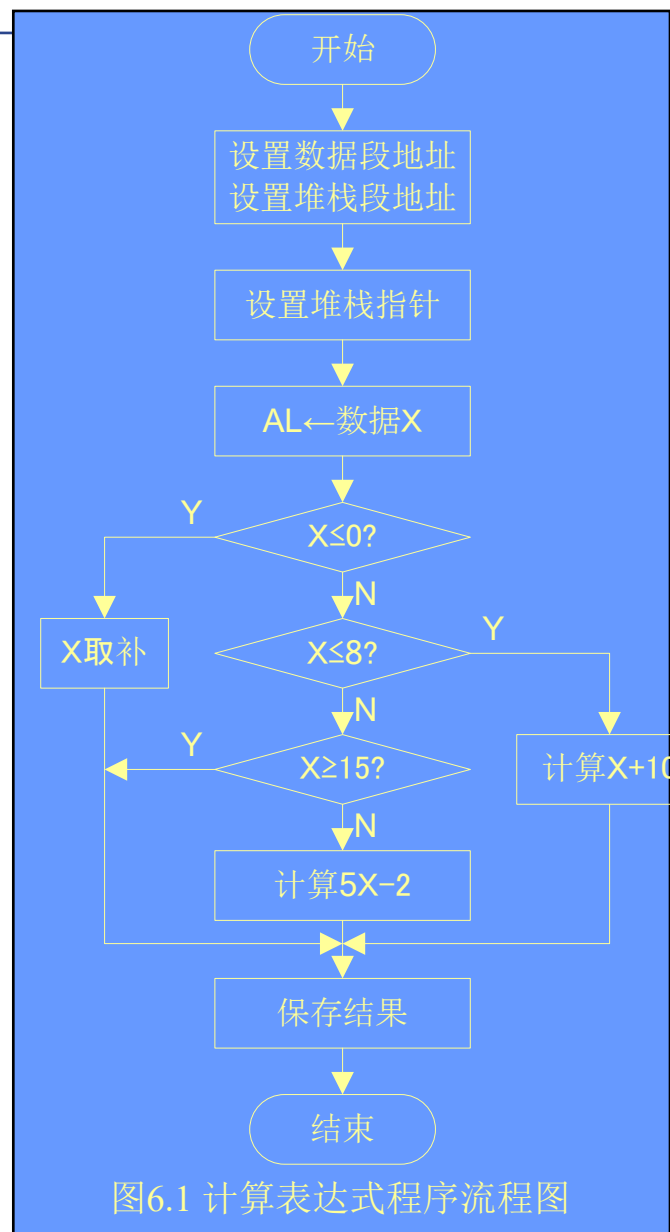


图6.1 计算表达式程序流程图

三. 分支结构程序设计

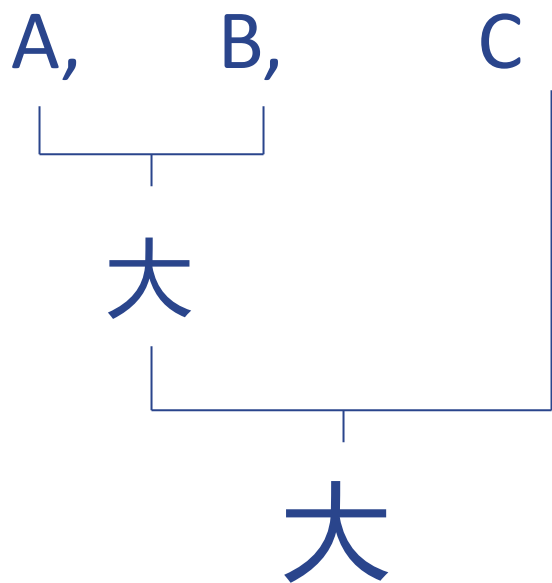
1	;*****EXAM6.1*****		21	JS	ABSL	;X<0转
2	✓ SSEG	SEGMENT STACK	22	JZ	MOVE	;X=0转
3	STK	DB 50DUP(0)	23	CMP	AL, 8	;X≤8?
4	SSEG	ENDS	24	JLE	ONE	;是,转
5			25	CMP	AL, 15	;X≥15?
6	✓ DSEG	SEGMENT	26	JGE	MOVE	;是,转
7	ARGX	DB -5	27	SAL	AL, 1	;计算5X-2
8	RLT	DB ?	28	SAL	AL, 1	
9	DSEG	ENDS	29	ADD	AL, ARGX	
10			30	SUB	AL, 2	
11	✓ CSEG	SEGMENT	31	JMP	MOVE	
12		ASSUME CS:CSEG, DS:DSEG	32	ONE:	ADD AL, 10	;X≤8,计算X+10
13		ASSUME SS:SSEG	33		JMP	MOVE
14	✓ BEGIN:	MOV AX, DSEG	34	ABSL:	NEG AL	;取补
15		MOV DS, AX	35	MOVE:	MOV RLT, AL	;保存结果
16		MOV AX, SSEG	36		MOV AH, 4CH	
17		MOV SS, AX	37		INT 21H	
18		MOV SP, SIZESTK	38	CSEG	ENDS	
19		MOV AL, ARGX	39	END	BEGIN	
20		AND AL, AL				

20 ;置标志位



三. 分支结构程序设计

▶ 例 6.2 设内存中有三个互不相等的无符号字数据，分别存放在ARG开始的字单元，编制程序将其中最大值存入MAX单元。



ARG+0	38H
+1	71H
+2	A6H
+3	84H
+4	9EH
+5	02H
MAX+0	
+1	

三. 分支结构程序设计

```
1  ;*****EXAM6.2*****
2  √ SSEG      SEGMENT STACK
3  STK         DB      20      DUP(0)
4  SSEG      ENDS
5
6  √ DSEG      SEGMENT
7  ARG         DW      7138H, 84A6H, 29EH
8  MAXDW      ?
9  DSEG      ENDS
10
11 √ CSEG      SEGMENT
12             ASSUME CS:CSEG, DS:DSEG
13             ASSUME SS:SSEG
14 √ FMAX:      MOV     AX, DSEG
15             MOV     DS, AX
16             MOV     AX, SSEG
17             MOV     SS, AX
18             MOV     SP, SIZESTK
19             LEA     SI, ARG      ;取数据首址
20             MOV     AX, [SI]    ;取第1个数
```

```
21             MOV     BX, [SI+2]  ;取第2个数
22             CMP     AX, BX      ;两数比较
23             JAE     FMAX1        ;AX中的数大
24             MOV     AX, BX      ;大数送AX
25 FMAX1:      CMP     AX, [SI+4]  ;大数与第3个数比较
26             JAE     FMAX2        ;AX中的数大
27             MOV     AX, [SI+4]  ;第3个数大大值
28 FMAX2:      MOV     MAX, AX     ;保存最大值
29             MOV     AH, 4CH
30             INT     21H
31 CSEG      ENDS
32 ENDF MAX
```

ARG+0	38H
+1	71H
+2	A6H
+3	84H
+4	9EH
+5	02H
MAX+0	
+1	

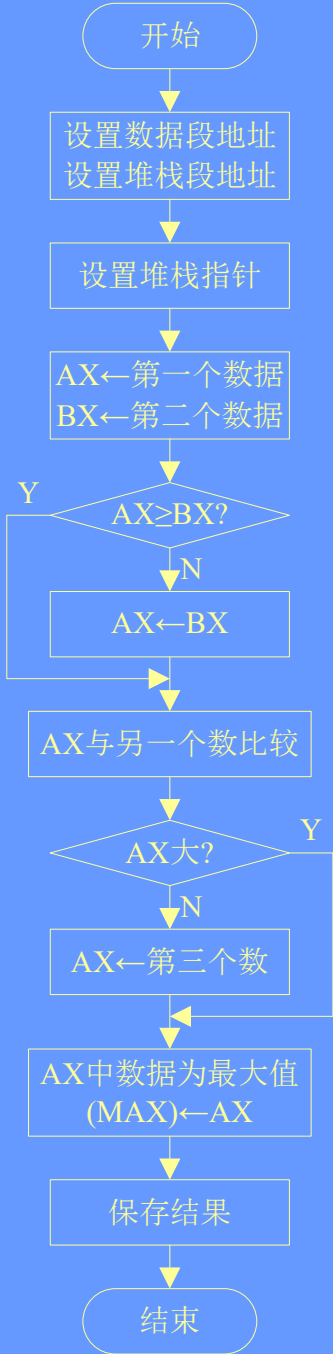


图6.2求最大值程序流程图

三. 分支结构程序设计

▶ 例 6.3 内存由ADR单元开始存放两个带符号字数据，编制程序，若两数同号将FLAG单元置0，否则置全1。

分析：判断两数是否同号，即判断两个数的最高位是否相同，若相同即为同号。判断的方法有两种：

第一种方法，先取出一个数，判断符号是否为正，若为正，再判断另一个数的符号是否为正，也为正，则两数同号，否则为异号；若第一个数的符号为负判断另一个数的符号是否为负，也为负，则两数同号，否则为异号。

三. 分支结构程序设计

1	;*****EXAM6.3.1*****		21	JNS	PLUS	;正转
2	SSEG	SEGMENT STACK	22	MOV	AX, ADR+2	
3	STK	DB 20 DUP(0)	23	AND	AX, AX	;第2个数置标志
4	SSEG	ENDS	24	JS	SAME	;同为负
5			25	USAM: MOV	AL, 0FFH	;异号标志
6	DSEG	SEGMENT	26	JMP	LOAD	
7	ADR	DW 73A5H, 924BH	27	PLUS: TEST	ADR+2, 8000H	;第2个数置标志
8	FLAG	DB ?	28	JS	USAM	;异号
9	DSEG	ENDS	29	SAME: XOR	AL, AL	;同号标志
10			30	LOAD: MOV	FLAG, AL	;存标志
11	CSEG	SEGMENT	31	MOV	AH, 4CH	
12		ASSUME CS:CSEG, DS:DSEG	32	INT	21H	
13		ASSUME SS:SSEG	33	CSEG	ENDS	
14	START:	MOV AX, DSEG	34	END	START	
15		MOV DS, AX				
16		MOV AX, SSEG				
17		MOV SS, AX				
18		MOV SP, SIZESTK				
19		MOV AX, ADR				
20		AND AX, AX				;置标志

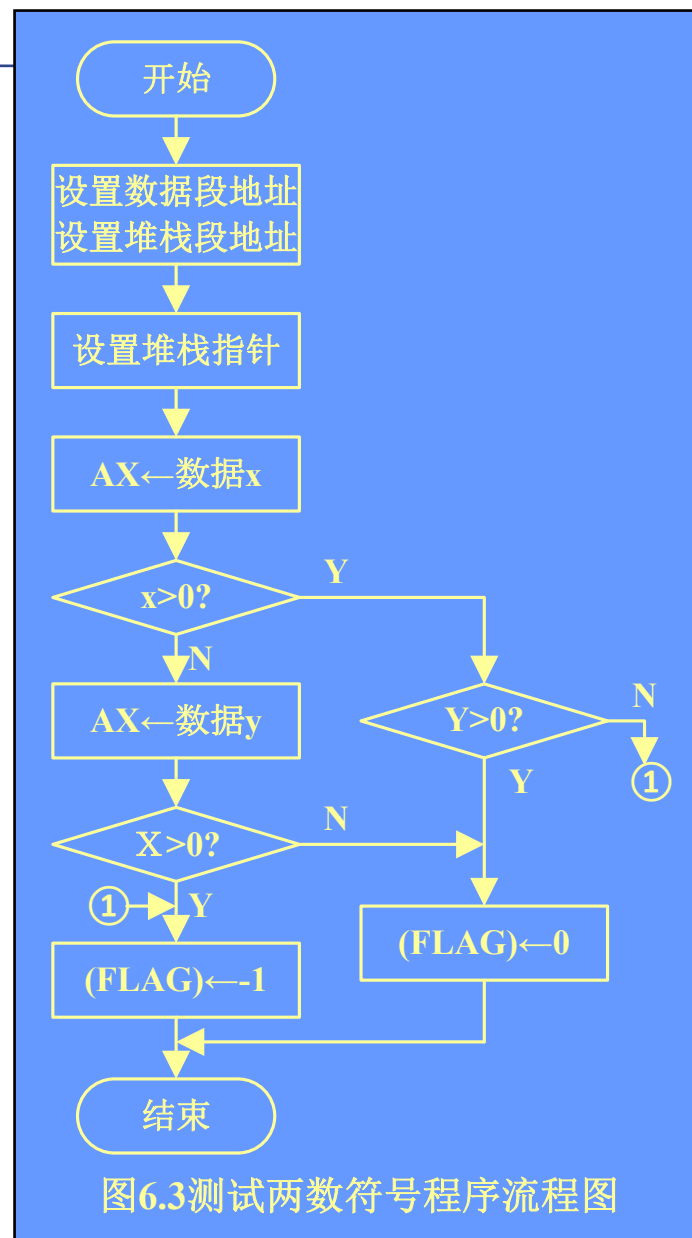


图6.3测试两数符号程序流程图

三. 分支结构程序设计

▶ 例 6.3 内存由ADR单元开始存放两个带符号字数据，编制程序，若两数同号将FLAG单元置0，否则置全1。

分析：判断两数是否同号，即判断两个数的最高位是否相同，若相同即为同号。判断的方法有两种：

第二种方法

0111 0011 1010 0101
^
1001 0010 0100 1011

1110 0001 1110 1110

三. 分支结构程序设计

```
1 ;*****EXAM6.3.1*****
2 SSEG SEGMENT STACK
3 STK DB 20 DUP(0)
4 SSEG ENDS
5
6 DSEG SEGMENT
7 ADR DW 73A5H,924BH
8 FLAG DB ?
9 DSEG ENDS
10
11 CSEG SEGMENT
12 ASSUME CS:CSEG, DS:DSEG
13 ASSUME SS:SSEG
14 START: MOV AX, DSEG
15 MOV DS, AX
16 MOV AX, SSEG
17 MOV SS, AX
18 MOV SP, SIZESTK
```

```
19 MOV AX, ADR
20 XOR AX, ADR+2
21 MOV AL, 0
22 JNS LOAD
23 DEC AL
24 LOAD: MOV FLAG, AL
25 MOV AH, 4CH
26 INT 21H
27 CSEG ENDS
28 END START
```

;两数异或
;同号标志
;同号
;异号标志
;存标志

ADR+0	A5H
+1	73H
+2	4BH
+3	92H

三. 分支结构程序设计

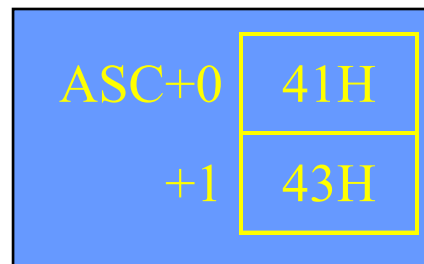
▶ 例 6.4 ASC单元存放两个字符的ASCII码，编程序检查其奇偶性，并将它们配制成奇校验存入原单元。

分析：字符的ASCII码是用七位二进制表示的，当用一个字节单元(8位)保存一个字符的ASCII码时，字节单元的第7位空闲，不同的计算机存放ASCII码时，系统软件对该位有不同的定义：

- 第7位总是0
- 第7位总是1
- 做为奇偶校验位
- 第7 位为1，扩充128种特殊字符或图形代码(在西文状态下)
- 做为汉字代码的标志位(在中文状态下)

三. 分支结构程序设计

```
1  ;*****EXAM6.4*****
2  SSEG      SEGMENT STACK
3  STK       DB      20  DUP(0)
4  SSEG      ENDS
5
6  DSEG      SEGMENT
7  ASC       DB      'AC'
8  DSEG      ENDS
9
10 CSEG      SEGMENT
11          ASSUME CS:CSEG, DS:DSEG
12          ASSUME SS:SSEG
13 START:    MOV     AX, DSEG
14          MOV     DS, AX
15          MOV     AX, SSEG
16          MOV     SS, AX
17          MOV     SP, SIZE STK
18          MOV     AX, WORD PTR ASC;取两字符
19          AND     AL, AL           ;置奇偶标志
20          JPO     NEXT           ;奇转
21          OR      AL, 80H         ;配为奇性
22 NEXT:     AND     AH, AH         ;置奇偶标志
23          JPO     LOAD           ;奇转
24          OR      AH, 80H         ;配为奇性
25 LOAD:     MOV     WORD PTR ASC, AX
26          MOV     AH, 4CH
27          INT     21H
28 CSEG      ENDS
29 END       START
```



四. 多分支结构程序设计

► 利用计算机解决实际问题时,常遇到这样的情况: 处理某个问题时有多种选择方案,根据实际情况选择其中一种。每种处理方案由一段程序完成,每一段程序可以看作一个分支,程序在执行过程中根据当前的状况,决定下一步应执行哪一个分支,这就构成了多个分支的程序。

如用计算机控制一台电动机,该电动机有正转、逆转,在每种转动方式下又有几种转速的档次控制,这些控制可以通过键盘0~9的数字键进行选择,进入某种档次选择后,执行相应分支程序,使电机以最佳方式由一个状态进入所选状态。假设程序的十个分支的起始地址分别为ADR0,ADR1,...ADR9。

四. 多分支结构程序设计

1: ;*****EXAM 6.5.1*****

2: SSEG SEGMENTSTACK

3: STK DB20 DUP(0)

4: SSEG ENDS

5: DSEG SEGMENT

6: DSEG ENDS

7: CSEG SEGMENT

8: ASSUME CS:CSEG,DS:DSEG

9: ASSUME SS:SSEG

10: MOTOR:MOV AX,DSEG

11: MOV DS,AX

12: MOV AX,SSEG

13: MOV SS,AX

14: MOV SP,SIZE STK

15: MOV AH,01

16: INT 21H

17: CMP AL,'0'

18: JZ ADR0

19: CMP AL,'1'

20: JZ ADR1

21: :

22: CMP AL,'8'

23: JZ ADR8

24:ADR9: :

25: :

: ADR0: :

: :

: ADR1: :

: :

:ADR8: :

:: :

: CSEG ENDS

: END MOTOR

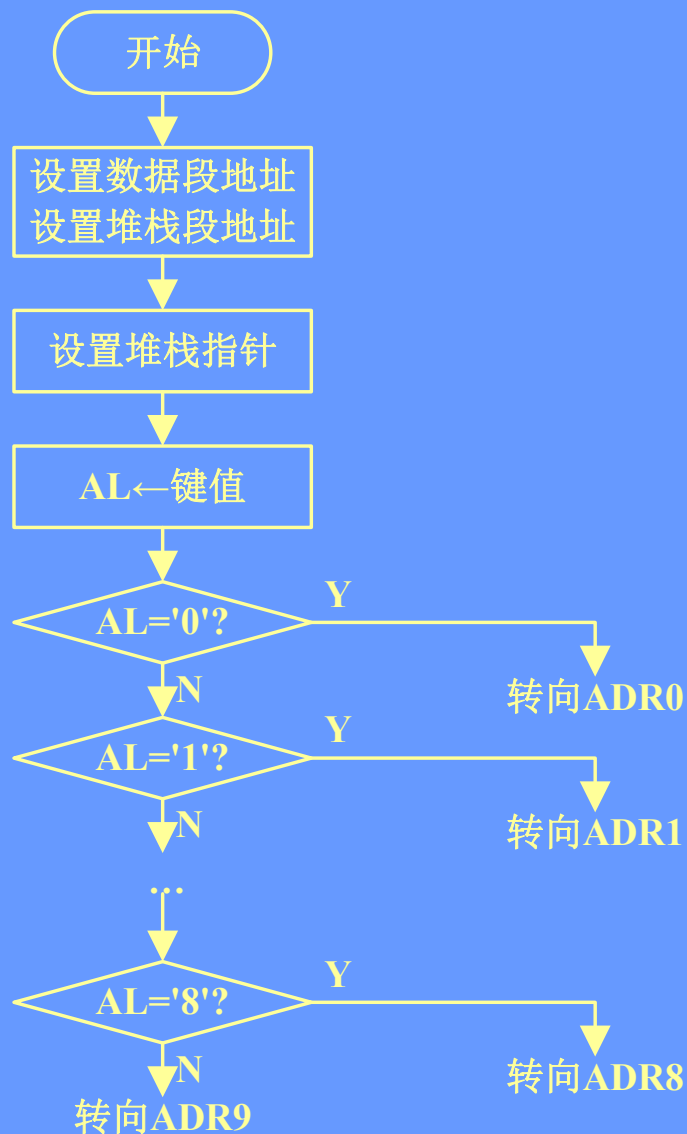
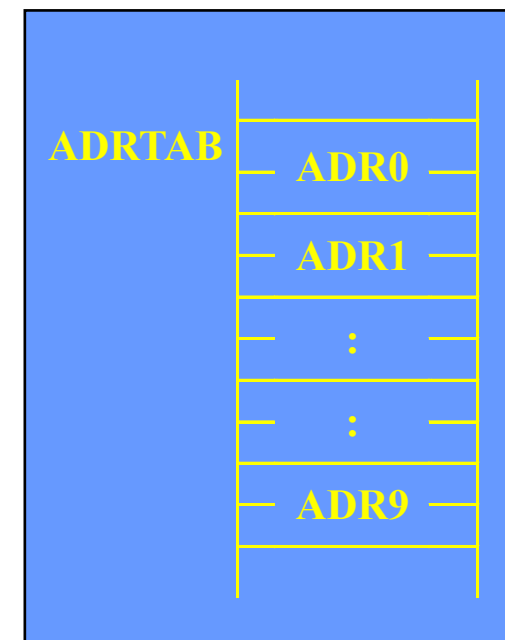


图6.5 树形分支结构流程图



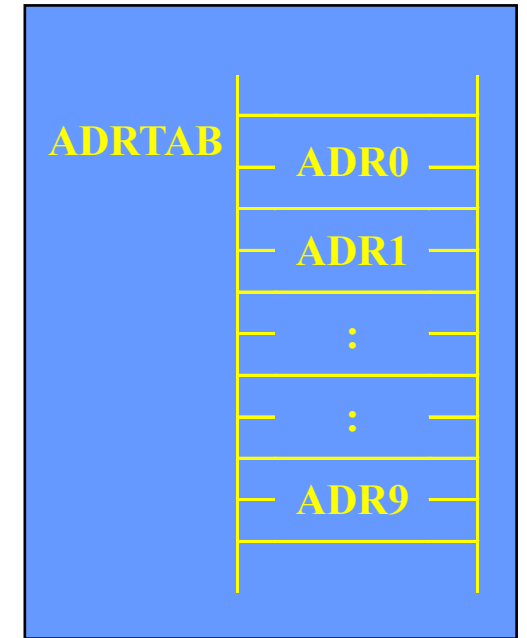
四. 多分支结构程序设计

► **地址常数表法**：所谓地址常数表法,就是把多个分支中的每个分支程序段的起始地址顺序存放在一个存储区中,这个存储区称地址表存储区。根据键值,将相应处理程序的入口地址取入某寄存器,然后用间接转移指令实现转移。



四. 多分支结构程序设计

```
1: ;*****EXAM 6.5.2 *****
2: SSEG      SEGMENT  STACK
3: STK       DB       20      DUP(0)
4: SSEG      ENDS
5: DSEG      SEGMENT
6: ADRTAB    DW       OFFSET ADR0,OFFSET ADR1
7:           DW       OFFSET ADR2,...,OFFSET ADR9
8:           DSEG      ENDS
9: CSEG      SEGMENT
10:          ASSUME    CS:CSEG,DS:DSEG
11:          ASSUME    SS:SSEG
12: BRANCH:MOV     AX,DSEG
13:           MOV      DS,AX
14:           MOV      AX,SSSEG
15:           MOV      SS,AX
16:           MOV      SP,LENGTH STK
```



四. 多分支结构程序设计

```
17:      LEA    DI,ADRTAB ;取地址表首址
18:      MOV    AH,01      ;读键盘
19:      INT     21H
20:      SUB     AL,'0'     ;转换为数字
21:      XOR     AH,AH      ;扩展为字数据
22:      SHL     AX,1       ;乘2
23:      ADD     DI,AX      ;形成相应地址
24:      MOV     AX,[DI]    ;取程序入口
25:      JMP     AX         ;转去执行
26:ADR0:  :
27:      :
   :ADR1:  :
   :      :
   :      :
   :ADR9:  :
   :      :
   :      :
:CSEG     ENDS
:         END    BRANCH
```

