



東北大學
Northeastern University

汇编语言程序设计

主讲：

刘松冉

单位：

东北大学计算机学院

智慧系统国际联合实验室

联系方式：liusongran@cse.neu.edu.cn

个人主页：<http://faculty.neu.edu.cn/liusongran>

<https://liusongran.github.io/>



随堂测试 – 第五章

- ▶ • 已知 : AX=1234H , BX=100H , CX=0A0F9H , SI=3405H , CF=1 ,
当前数据段中 (3305H) =13H , (3306H) =58H , (3405H)
=56H , (3406H) =41H , (3505H) =6235H , DX=4389H ,
指出执行下述各条指令后, 目标地址中的数值。

- (1) MOV AX, BX (2) MOV DL, CH
(3) MOV AX, [SI] (4) MOV AL, [SI+1]
(5) MOV BX, [SI-100H] (6) MOV DX, -200H[BX][SI]
(7) MOV [BX][SI], CX (8) MOV [BX], AX

3305H	13H
3306H	58H
...	...
3405H	56H
3406H	41H
...	...
3505H	35H
3506H	62H

1. 编写完成下列功能的小程序

(1) 将寄存器AL的第3, 4位变反, 其余位保持不变。

解: XOR AL, 18H

(2) 将寄存器AL的第3, 4位清零, 其余位保持不变。

解: AND AL, 0E7H

(3) 将寄存器AL的第3, 4位置1, 其余位保持不变。

解: OR AL, 18H

(4) DX和AX联合存放一个32位整数(DX存放高16位), 将该数据逻辑左移一位。

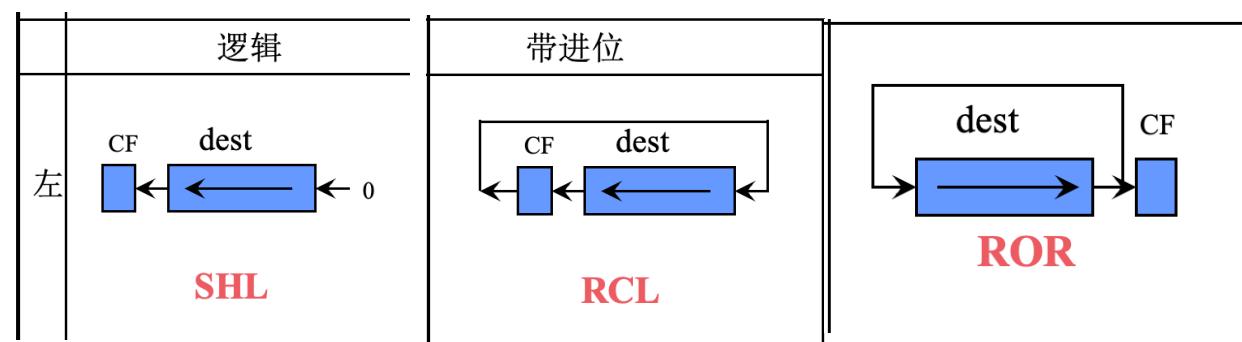
解: SHL AX, 1

RCL DX, 1

(5) 内存单元MEMB存放着一个8位的数据, 将它的高4位和低4位互换。

解: MOV CL, 4

ROR MEMB, CL



2. 编写程序将AL中的第7位和第0位，第6位和第1位，第5位和第2位，第4位和第3位互换。

CF AL
B₇ B₆B₅B₄B₃B₂B₁B₀

AH
B₇X X X X X X X

CF AL
B₆ B₅B₄B₃B₂B₁B₀00

AH
B₆B₇X X X X X X X

解: SHL AL,1

RCR AH,1

MOV AL,AH

- |▶ 3. 编程序将AL中的第i，i+1位写入MEM单元的第i，i+1位，其他内容不变。

MOV CL, i

MOV BL, 03H

SHL BL, CL

AND AL, BL

NOT BL

AND MEM, BL

OR MEM, AL

4. 对于下述指令，当AX=8573H，BX=8032H时，写出各指令执行后标志位的状态变化及目标地址内容。

指 令	OF	SF	ZF	PF	CF	目标地址内容
(1) ADD AL, BL	1	1	0	1	0	0A5H
(2) SUB AH, BL	1	0	0	1	0	53H
(3) NEG BH	1	1	0	0	1	80H
(4) SAR AH, 1	0	1	0	0	1	0C2H
(5) AND AL, BL	0	0	0	0	0	32H

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

|> 从第一条指令开始，逐条顺序执行每一条指令，直到遇停机指令为止，称为顺序结构。

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：

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

一. 分支结构程序的引出

- • 必须把两种处理方法编制为两段程序都存入计算机。根据行李的重量决定执行哪一段程序
- 程序有两个分支，形成了分支结构。计算机根据判断结果执行其中的一个分支
- 计算机进行逻辑判断，有真、假两种判断结果
- 根据真假执行两个不同程序段的基本方法：
 - 使用条件转移指令
 - 当条件满足时，转向指定的标号执行程序
 - 否则，顺序执行程序

一. 分支结构程序的引出

► 条件转移指令可以测试下述条件：

- 大于、大于等于、等于、不等于、小于、小于等于、高于、高于等于、低于、低于等于、溢出、未溢出、正、负、奇和偶等

```
:  
    CMP AX, BX      ;两数比较  
    JB  MOVE        ;AX中数小时,转 MOVE  
    MOV AX, BX      ;将小数放入 AX  
MOVE:  MOV MIN, AX  ;小数存入 MIN 单元  
    :  
:
```

```
    AND AX, AX  
    JS   MOVE       ;测试 AX 的符号,若为负,转 MOVE  
    MOV PLUS, AX    ;AX中为正数  
    :  
MOVE:  MOV MINUS, AX  ;AX中为负数  
    :  
:
```

```
    MOV CX, 00  
    MOV FLAG, CX    ;预置 FLAG 为 0  
    ADD AX, BX      ;两数相加  
    JNO MOVE        ;若无溢出,转 MOVE  
    DEC WORD PTR FLAG  ;将 FLAG 变为全 1  
MOVE:  MOV SUM, AX  
    :  
:
```

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：

$$A < B \qquad A \geq B$$

$$A \leq B \qquad A = B$$

$$A > B \qquad A \neq B$$

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^OF)=1
A ≤ B	JBE	CF=1 ZF=1	JLE	(SF^OF) ZF=1
A > B	JA	CF=0&ZF=0	JG	(SF^OF) ZF=0
A ≥ B	JAE	CF=0	JGE	(SF^OF)=0
A = B		JE (ZF=1)		
A ≠ B		JNE (ZF=0)		

Below Above Less Great Equal

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^OF)=1
A ≤ B	JBE	CF=1 ZF=1	JLE	(SF^OF) ZF=1
A > B	JA	CF=0&ZF=0	JG	(SF^OF) ZF=0
A ≥ B	JAE	CF=0	JGE	(SF^OF)=0
A = B		JE (ZF=1)		
A ≠ B		JNE (ZF=0)		

JNAE JNA JNBE JNB JNGE JNG JNLE JNL

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。

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

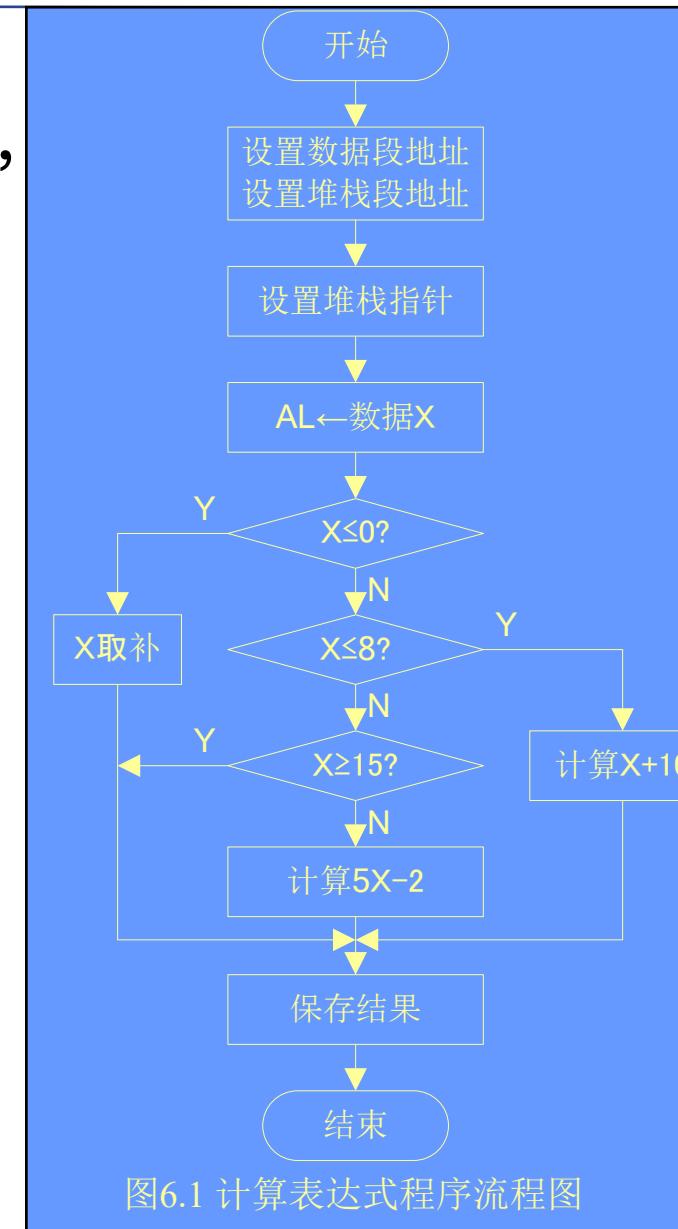


图6.1 计算表达式程序流程图

三. 分支结构程序设计

```

1 ;*****EXAM6.1*****
2 √ SSEG SEGMENT STACK
3 STK DB 50DUP(0)
4 SSEG ENDS
5
6 √ DSEG SEGMENT
7 ARGX DB -5
8 RLT DB ?
9 DSEG ENDS
10
11 √ CSEG SEGMENT
12 ASSUME CS:CSEG, DS:DSEG
13 ASSUME SS:SSEG
14 √ BEGIN: MOV AX, DSEG
15 MOV DS, AX
16 MOV AX, SSEG
17 MOV SS, AX
18 MOV SP, SIZESSTK
19 MOV AL, ARGX
20 AND AL, AL
;取X
;置标志位

```

	JS	ABSL	;X<0转
	JZ	MOVE	;X=0转
21	CMP	AL, 8	;X≤8?
22	JLE	ONE	;是,转
23	CMP	AL, 15	;X≥15?
24	JGE	MOVE	;是,转
25	SAL	AL, 1	;计算5X-2
26	SAL	AL, 1	
27	ADD	AL, ARGX	
28	SUB	AL, 2	
29	JMP	MOVE	
30	ONE:	ADD AL, 10	;X≤8,计算X+10
31		JMP MOVE	
32	ABSL:	NEG AL	;取补
33	MOVE:	MOV RLT, AL	;保存结果
34		MOV AH, 4CH	
35		INT 21H	
36	CSEG	ENDS	
37			
38			
39	END BEGIN		

		无符号数		带符号数	
A<B	JB	CF=1	JL	(SF^OF)=1	
A≤B	JBE	CF=1 ZF=1	JLE	(SF^OF) ZF=1	
A>B	JA	CF=0&ZF=0	JG	(SF^OF) ZF=0	
A≥B	JAE	CF=0	JGE	(SF^OF)=0	
A=B		JE (ZF=1)			
A≠B		JNE (ZF=0)			

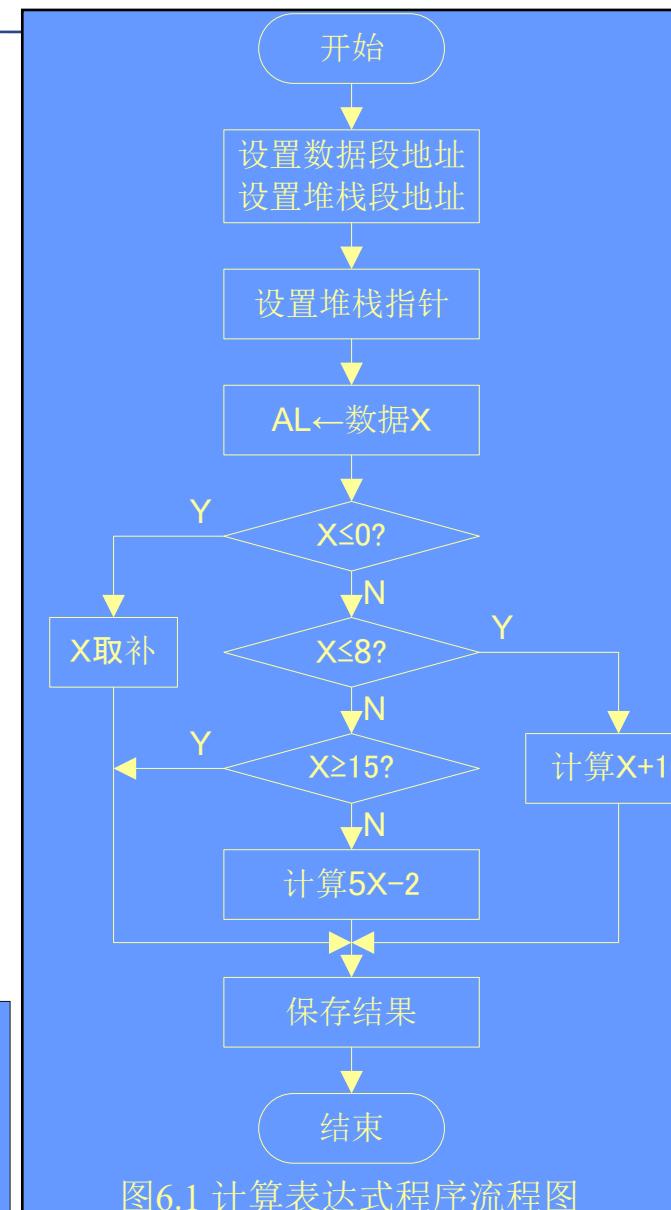


图6.1 计算表达式程序流程图

三. 分支结构程序设计

```

1 ;*****EXAM6.1*****
2 √ SSEG SEGMENT STACK
3 STK DB 50DUP(0)
4 SSEG ENDS
5
6 √ DSEG SEGMENT
7 ARGX DB -5
8 RLT DB ?
9 DSEG ENDS
10
11 √ CSEG SEGMENT
12 ASSUME CS:CSEG, DS:DSEG
13 ASSUME SS:SSEG
14 √ BEGIN: MOV AX, DSEG
15 MOV DS, AX
16 MOV AX, SSEG
17 MOV SS, AX
18 MOV SP, SIZESTK
19 MOV AL, ARGX
20 AND AL, AL
    
```

	JS	ABSL	;X<0转
21	JZ	MOVE	;X=0转
22	CMP	AL, 8	;X≤8?
23	JLE	ONE	;是,转
24	CMP	AL, 15	;X≥15?
25	JGE	MOVE	;是,转
26	SAL	AL, 1	;计算5X-2
27	SAL	AL, 1	
28	ADD	AL, ARGX	
29	SUB	AL, 2	
30	JMP	MOVE	
31	ONE:	ADD AL, 10	;X≤8,计算X+10
32		JMP MOVE	
33	ABSL:	NEG AL	;取补
34	MOVE:	MOV RLT, AL	;保存结果
35		MOV AH, 4CH	
36		INT 21H	
37	CSEG	ENDS	
38			
39	END BEGIN		

数据不变
影响标志

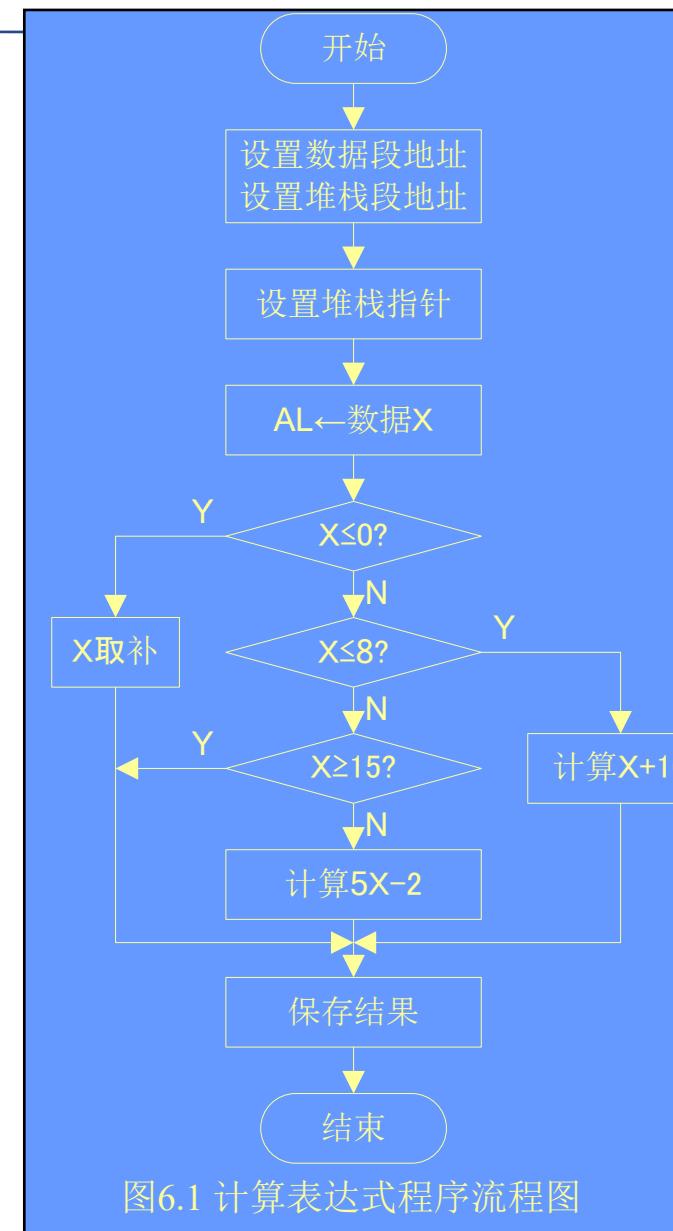
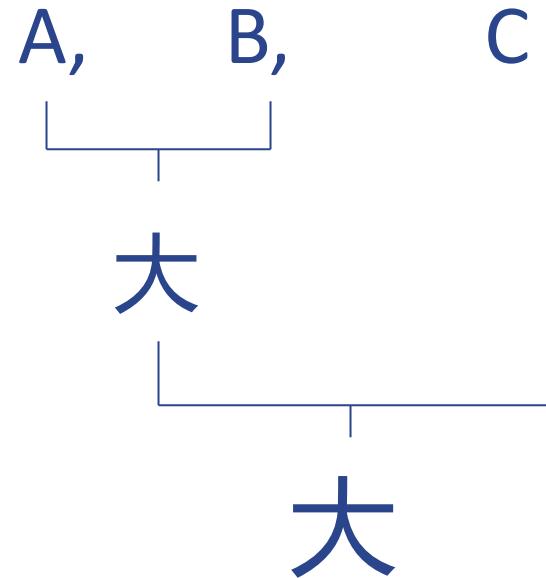


图6.1 计算表达式程序流程图

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	(空)
+1	(空)

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

```

A ≥ B | JAE | CF = 0

	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
FMAX1:	25	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
FMAX2:	28	MOV MAX, AX	;保存最大值
	29	MOV AH, 4CH	
	30	INT 21H	
CSEG	31	ENDS	
ENDF MAX	32		

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

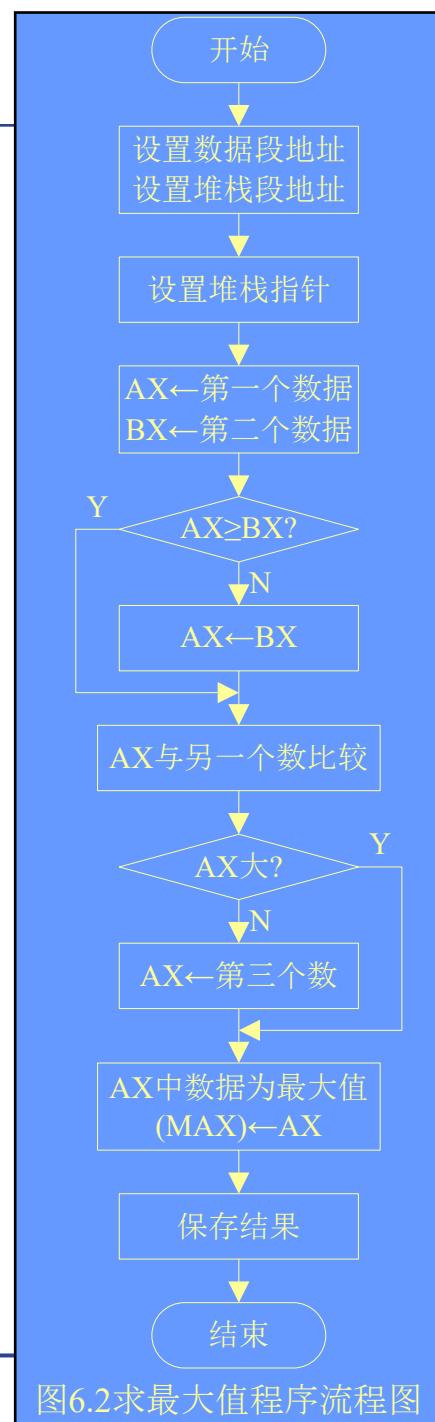


图6.2求最大值程序流程图

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。

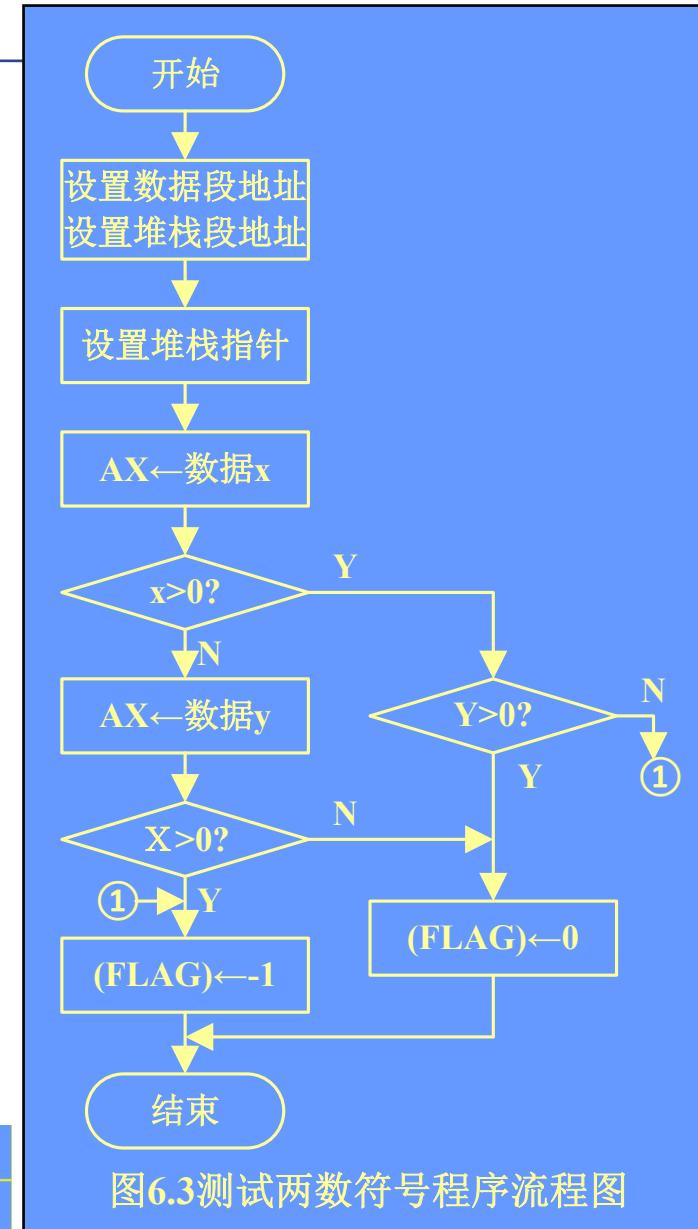
三. 分支结构程序设计

```

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

```

SF	JS	lab	SF=1
	JNS	lab	SF=0



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
20
21
22
23
24     LOAD: MOV AX, ADR
25             XOR AX, ADR+2 ;两数异或
26             MOV AL, 0 ;同号标志
27             JNS LOAD ;同号
28             DEC AL ;异号标志
29             MOV FLAG, AL ;存标志
30             MOV AH, 4CH
31             INT 21H
32
33         CSEG ENDS
34
35     END START
```

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，编制程序检查其奇偶性，并将它们配制成为奇校验存入原单元。

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)

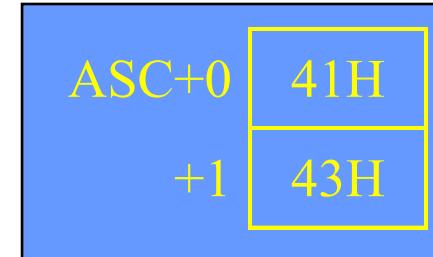
三. 分支结构程序设计

1 ;*****EXAM6.4*****

2 SSEG SEGMENT STACK
 3 STK DB 20 DUP(0)
 4 SSEG ENDS

PF	JP/JPE	lab	PF=1	Even
	JNP/JPO	lab	PF=0	

Odd



DSEG	SEGMENT	18	MOV	AX, WORD PTR ASC;取两字符
ASC	DB 'AC'	19	AND	AL, AL ;置奇偶标志
DSEG	ENDS	20	JPO	NEXT ;奇转
		21	OR	AL, 80H ;配为奇性
CSEG	SEGMENT	22	NEXT:	AND AH, AH ;置奇偶标志
	ASSUME CS:CSEG, DS:DSEG	23	JPO	LOAD ;奇转
	ASSUME SS:SSEG	24	OR	AH, 80H ;配为奇性
START:	MOV AX, DSEG	25	LOAD:	MOV WORD PTR ASC, AX
	MOV DS, AX	26	MOV	AH, 4CH
	MOV AX, SSEG	27	INT	21H
	MOV SS, AX	28	CSEG	ENDS
	MOV SP, SIZE STK	29	END	START

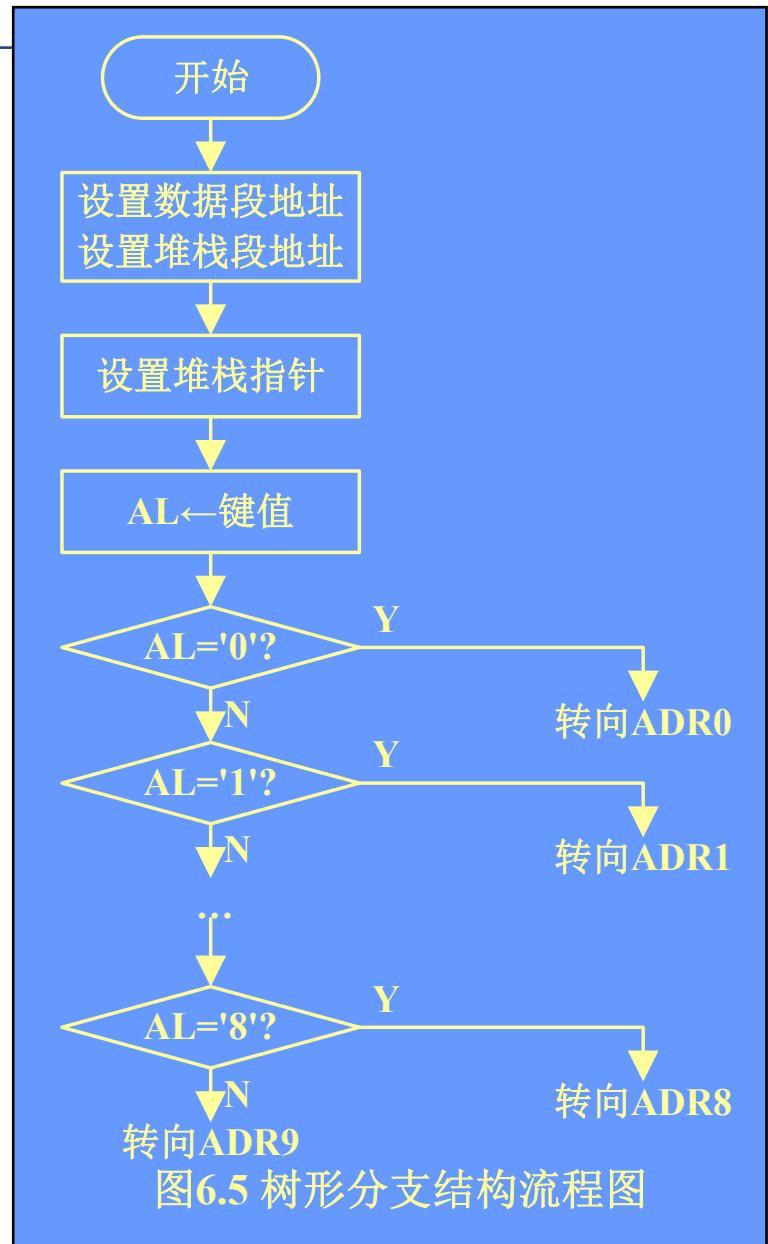
四. 多分支结构程序设计

▶ 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。

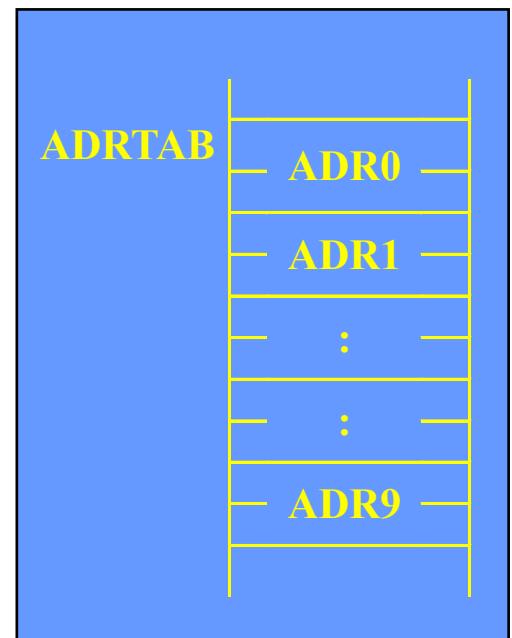
四. 多分支结构程序设计

```
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
```



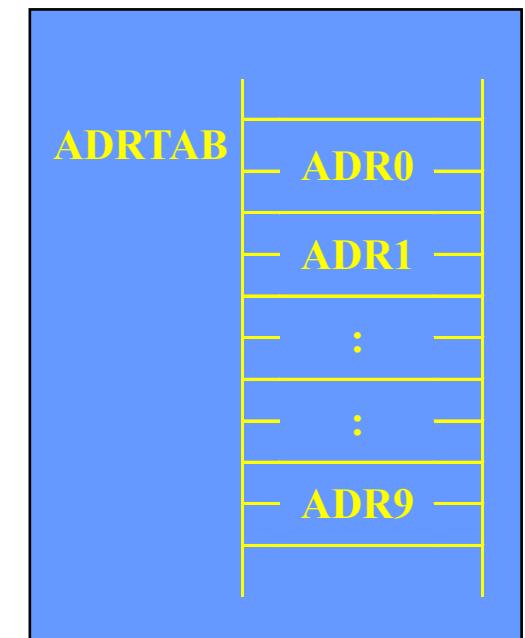
四. 多分支结构程序设计

► **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。



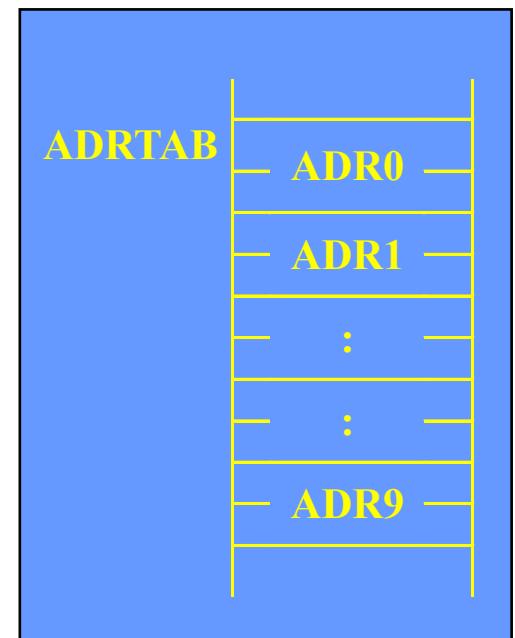
四. 多分支结构程序设计

```
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,SSEG
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
```



课后作业 – 第六章

1. 已知AX=7380H，BX=2500H，写出下述每条指令执行后OF，SF，ZF，PF，CF的状态和目标地址内容。

指 令	OF	SF	ZF	PF	CF	目标地址内容
(1) ADD AX, BX						
(2) SUB AL, AH						
(3) CMP AL, BH						
(4) NEG BL						
(5) AND AL, BH						
(6) OR AL, AH						
(7) SHL AL, 1						

指 令	OF	SF	ZF	PF	CF	目标地址内容
(15) SUB AL, BH						
(16) MUL BH						
(17) IMUL BX						
(18) XOR BH, AH						
(19) NEG AL						
(20) SHR AH, 1						
(21) ROR AH, 1						

指 令	OF	SF	ZF	PF	CF	目标地址内容
(8) SHL BL, 1						
(9) SAR AL, 1						
(10) ROL AH, 1						
(11) ROL AL, 1						
(12) ROR BH, 1						
(13) ADD AH, AH						
(14) ADD AH, BH						

2. 下列程序中有错误，请把他们找出来。已知程序欲完成的功能是从两个无符号数中选出较大者送入MAX单元，若二者相同，将MAX清0。

```
1. SEGA SEGMENT  
2. DAX: DW 200  
3. DAY DB 1000H  
4. MAX DB 0,0  
5. SEGA ENDS  
6. CSEG SEGMENT  
7. ASSUME CS:CSEG  
8. ASSUME DS:SEGA  
9. CMP AX, DAX  
10. JG XGY  
11. JZ ZERO  
12. MOV AX, DAY  
13. XGY: MOV MAX, AX  
14. ZERO HLT  
15. ENDS  
16. END START
```

3. 下面一段信息是用DEBUG命令的结果，填充空白处的内容。

C>DEBUG EXAM.EXE

-R

AX=023A BX=0000 CX=0045 DX=0000 SP=FFFE BP=0000 SI=0000
DI=0000 DS=0913 ES=0913 SS=0913 CS=0913 IP=0000.....

-E DS:0

0913:0000 5 96 46 57 00 45

-U3

0913:0003 BE 00 00 MOV SI, 00
0913:0006 8A 04 MOV AL, [SI]
0913:0008 02 44 01 ADD AL, [SI+1]
0913:000B 79 06 JNS 13
0913:000D F6 D8 NEG AL
0913:000F 88 44 02 MOV [SI+2], AL
0913:0012 CC INT 3
0913:0013

-G=3 13

AX=_____ BX=_____ SI=_____

D DS:0 7

0913 : 0000 _____

4. 下面一段信息是用DEBUG命令的结果，填充空白处的内容。

C>DEBUG EXAM.EXE

-U 5

```
0933:0005 BB 01 00      MOV BX, 0001
0933:0008 B0 24        MOV AL, 24
0933:000A 74 12        JZ  001E
0933:000C 8D 1E 06 01   LEA BX, [0106]
0933:0010 8A 07        MOV AL, [BX]
0933:0012 00 47 01      ADD [BX+01], AL
0933:0015 70 07        JO   001E
0933:0017 28 47 02      SUB [BX+02], AL
0933:001A F6 E1        MUL CL
0933:001C 73 03        JNB 0021
0933:001E 88 47 02      MOV [BX+2], AL
0933:0021 89 47 02      MOV [BX+2], AX
0933:0024 CC            INT 3
0933:0025 .....
```

-R

```
AX=1234  BX=0123  CX=01FF  DX=0100  SP=FFFE  BP=0000  SI=0000
DI=1200  DS=0923  ES=0923  SS=0923  CS=0933  IP=000C
```

.....

-G 1A 21

AX=_____ BX=_____
-D 120 124