Assembly Programming

第十周 循环与分枝程序设计

庞彦

yanpang@gzhu.edu.cn

Fall 2023

- 控制转移指令
- 循环结构程序设计
- 分支结构程序设计

- 控制转移指令
- 循环结构程序设计
- 分支结构程序设计

程序设计基本步骤

一般说来，编制一个汇编语言程序需要完成以下步骤：

(1) 分析题意，建立数学模型，确定数据结构及算法。这一步是能否编制出高质量程序的关键，因此不应该一拿到题目就急于写程序，而是应该仔细地分析和理解题意，找出合理的算法及适当的数据结构。

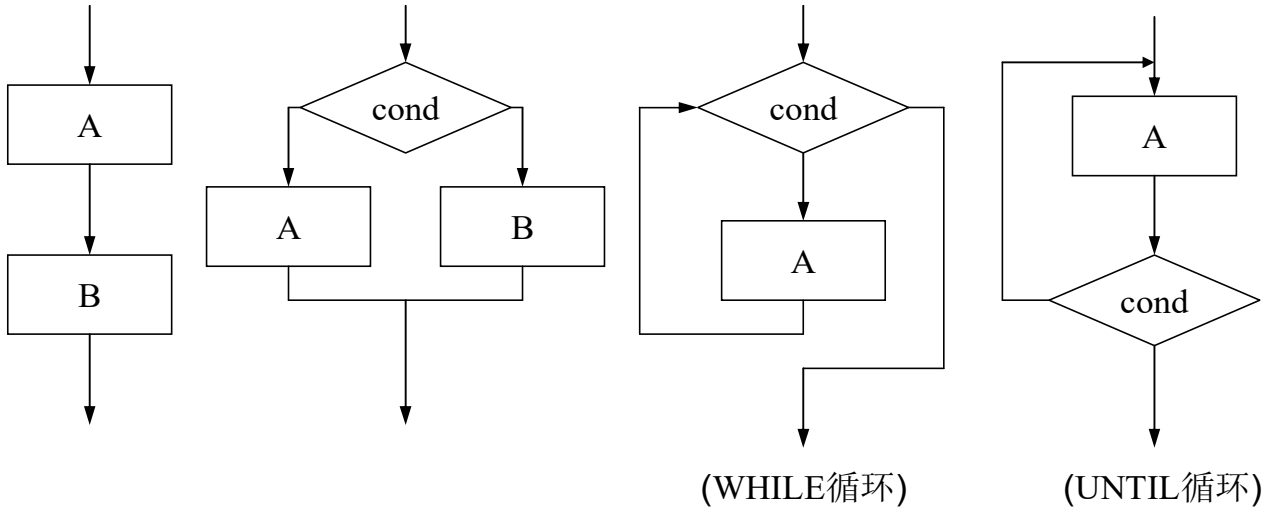
(2) 根据算法画出程序流程图。这一步对初学者尤其重要，这样做可以减少出错的可能性。画流程图时可以从粗到细把算法逐步地具体化。

(3) 编写汇编语言源程序，根据算法及数据结构分配内存单元和寄存器。

(4) 使用汇编程序调试工具上机调试程序。

程序设计基本步骤

三种程序构件



(a) 顺序结构　　(b) 分支结构　　(c) 循环结构

控制转移指令

控制转移指令:

- 无条件转移指令

  JMP

- 条件转移指令

  JZ / JNZ 、 JE / JNE、 JS / JNS、 JO / JNO、

  JP / JNP、 JB / JNB、 JL / JNL、 JBE / JNBE、

  JLE / JNLE、 JCXZ

- 循环指令

  LOOP、LOOPZ / LOOPE、LOOPNZ / LOOPNE

控制转移指令

无条件转移指令：

段内直接短转移：JMP SHORT OPR

执行操作：(IP) ← (IP) + 8位位移量

段内直接近转移：JMP NEAR PTR OPR

执行操作：(IP) ← (IP) + 16位位移量

段内间接转移：JMP WORD PTR OPR

执行操作：(IP) ← (EA)

段间直接远转移：JMP FAR PTR OPR

执行操作：(IP) ← OPR 的段内偏移地址

(CS) ← OPR 所在段的段地址

段间间接转移：JMP DWORD PTR OPR

执行操作：(IP) ← (EA)

(CS) ← (EA+2)

控制转移指令

条件转移指令：只能使用段内直接寻址的8 位位移量（386以后机型支持16位位移量）

(1) 根据单个条件标志的设置情况转移

指令的助忆符	检测的转移条件	功能描述
JZ/JE	ZF=1	结果为0或相等则转移
JNZ/JNE	ZF=0	结果不为0或不相等则转移
JS	SF=1	结果为负则转移
JNS	SF=0	结果不为负则转移
JO	OF=1	结果溢出则转移
JNO	OF=0	结果不溢出则转移
JP/JPE	PF=1	奇偶位为1则转移(1的个数为偶数时为1)
JNP/JPO	PF=0	奇偶位为0则转移
JB/JNAE/JC	CF=1	低于或不高于或等于，或CF=1则转移
JNB/JAE/JNC	CF=0	不低于或高于或等于，或CF=0则转移

控制转移指令

(2) 比较两个无符号数，并根据比较结果转移*

两数的高低分成4种关系：

(1) <，低于（不高于等于）：JB（JNAE），CF=1

(2) ≥，不低于（高于等于）：JNB（JAE），CF=0

(3) ≤，低于等于（不高于）：JBE（JNA），CF∨ZF=1

(4) >，不低于等于（高于）：JNBE（JA），CF∨ZF=0

* 适用于地址或双精度数低位字的比较

控制转移指令

(3) 测试 CX 的值为 0 则转移

格式	测试条件
JCXZ OPR	(CX)=0

(4) 比较两个带符号数，并根据比较结果转移*

	格式	测试条件
<	JL (JNGE) OPR	SF∀OF = 1 （异或）
≥	JNL (JGE) OPR	SF∀OF = 0
≤	JLE (JNG) OPR	(SF∀OF)∨ZF = 1
>	JNLE (JG) OPR	(SF∀OF)∨ZF = 0

* 适用于带符号数的比较

控制转移指令

<	JL (JNGE)	OPR	SF∀OF = 1

带符号数比较结果的几种情况：

两个异号数相加或同号数相减，结果不会溢出。

两个同号数相加或异号数相减，有可能溢出。

正溢出：结果大于机器能表示的最大正数

负溢出：结果小于机器能表示的最小负数

例：比较两个数，若A<B，则转到label去执行。

控制转移指令

< JL (JNGE) OPR SF∀OF = 1

例：比较两个数，若A<B，则转到label去执行。

```
MOV AX, A
CMP AX, B
JL label
```

带符号数比较结果的几种情况：

两个异号数相加或同号数相减，结果不会溢出。

两个同号数相加或异号数相减，有可能溢出。

正溢出：结果大于机器能表示的最大正数

负溢出：结果小于机器能表示的最小负数

1. 若A-B的结果使得SF=0，OF=0，说明差值为正，且未溢出，可以判断A≥B，SF∀OF = 0，不满足转移条件。（例：比较20, 8）
2. 若A-B的结果使得SF=0，OF=1，说明差值为正，且溢出，这种情况必为负溢出，可以判断A<B，SF∀OF =1，满足转移条件。
3. 若A-B的结果使得SF=1，OF=0，说明差值为负，且未溢出，可以判断A<B，SF∀OF = 1，满足转移条件。（例：比较8, 20）
4. 若A-B的结果使得SF=1，OF=1，说明差值为负，且溢出，这种情况必为正溢出，可以判断A>B，SF∀OF = 0，不满足转移条件。

控制转移指令

< JL (JNGE) OPR SF∀OF = 1

正溢出：结果大于机器能表示的最大正数

负溢出：结果小于机器能表示的最小负数

若A-B的结果使得SF=0，OF=1，说明差值为正，且溢出，这种情况必为负溢出，可以判断A<B，SF∀OF =1，满足转移条件。
若A-B的结果使得SF=1，OF=1，说明差值为负，且溢出，这种情况必为正溢出，可以判断A>B，SF∀OF = 0，不满足转移条件。

正溢出：89-(-108)

```
  01011001（89）
  01101100（108）
------------------------
  11000101（-59）
```

SF=1，OF=1
A>B，不满足转移条件

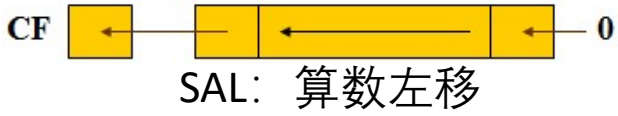
负溢出：-110-92

```
        10010010（-110）
        10100100（-92）
------------------------
    1   00110110（54）
```

SF=0，OF=1
A<B，满足转移条件

控制转移指令

例：统计AX寄存器中为1位数的，并将统计结果放在CL寄存器中。



SAL：算数左移

```
        MOV   CL，0        ；置循环初值
        MOV   BX，16
LAB1: SAL     AX, 1        ；将AX的内容左移一位，即最高位移到CF
        JNC   LAB2         ；如果CF=0则表示AX的最高位为0，转LAB2
        INC   CL           ；如果CF=1则表示AX的最高位为1，个数加1
LAB2: DEC     BX           ；修改循环次数，未完则转LAB1
        JNZ   LAB1
EXIT:…
```

```
        MOV     CL,0
LAB:  AND       AX，AX
        JZ      EXIT        ；AX=0时循环结束，转到EXIT
        SAL     AX，1       ；将AX中的最高位移入CF中
        JNC     LAB         ；如果CF=0则转LAB
        INC     CL          ；如果CF=1则CL+1→CL
        JMP     LAB         ；转LAB处继续循环
EXIT:  …
```

控制转移指令

例：α、β 是带符号双精度数，分别存于DX, AX及BX, CX中，α > β 时转 L1，否则转 L2。

控制转移指令

例：α、β 是带符号双精度数，分别存于DX, AX及BX, CX中，α > β 时转 L1，否则转 L2。

```
        CMP   DX, BX
        JG    L1
        JL    L2
        CMP   AX, CX
        JA    L1
L2:
        ……

L1:
        ……
```

控制转移指令

例：α、β 是带符号双精度数，分别存于DX, AX及BX, CX中，α > β 时转 L1，否则转 L2。

```
        CMP   DX, BX
        JG    L1
        JL    L2
        CMP   AX, CX
        JA    L1
L2:
        ……
        JMP L3
L1:
        ……
L3:
        ……
```

重要!

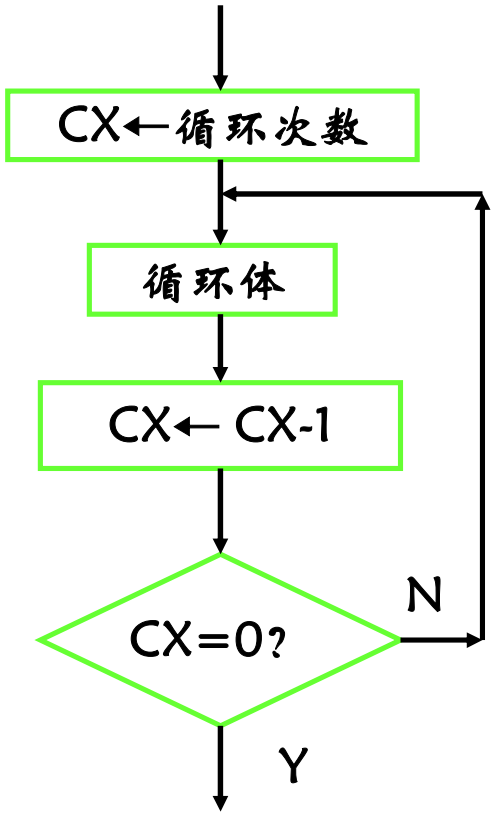
控制转移指令

循环指令 LOOP

语句格式： LOOP 短标号

执行过程：

1. (CX)＝(CX)-1（不改变任何标志位）
2. 如果(CX)≠0，转向“标号”所指向的指令；否则，终止循环，执行该指令下面的指令。



控制转移指令

例：求首地址为 ARRAY 的 M 个字之和（不考虑溢出），结果存入 TOTAL中。

```
        MOV    CX, M              ;循环次数
        MOV    AX, 0
        MOV    SI, 0
AGAIN:
```

控制转移指令

例：求首地址为 ARRAY 的 M 个字之和（不考虑溢出），结果存入 TOTAL中。

```
        MOV   CX, M           ;循环次数
        MOV   AX, 0
        MOV   SI, 0
AGAIN:
        ADD   AX, ARRAY[SI]
        ADD   SI, 2
        LOOP  AGAIN
        MOV   TOTAL, AX
```

控制转移指令

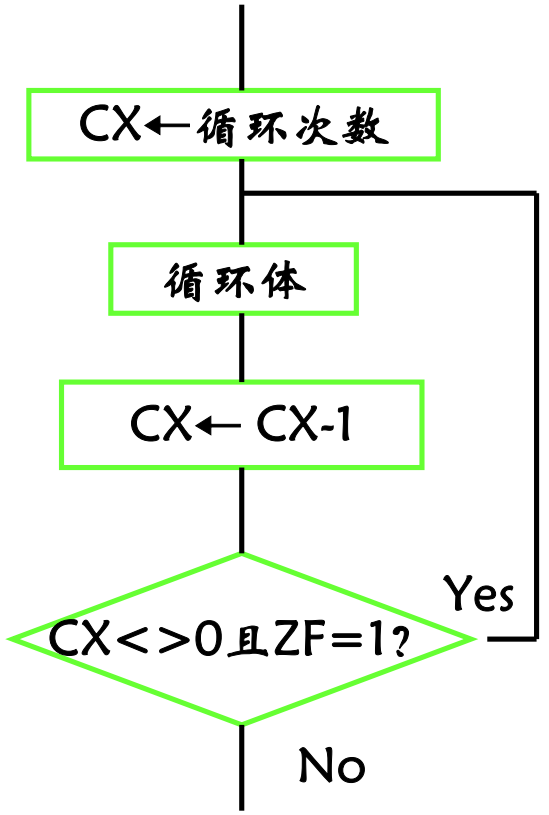
相等/为零循环指令 LOOPE/LOOPZ

语句格式： LOOPE/ LOOPZ 短标号

执行过程：

1. (CX)＝(CX)-1（不改变任何标志位）

2. 如果CX≠0且ZF=1，则程序转到循环体的第一条指令；否则，程序将执行该循环指令下面的指令。



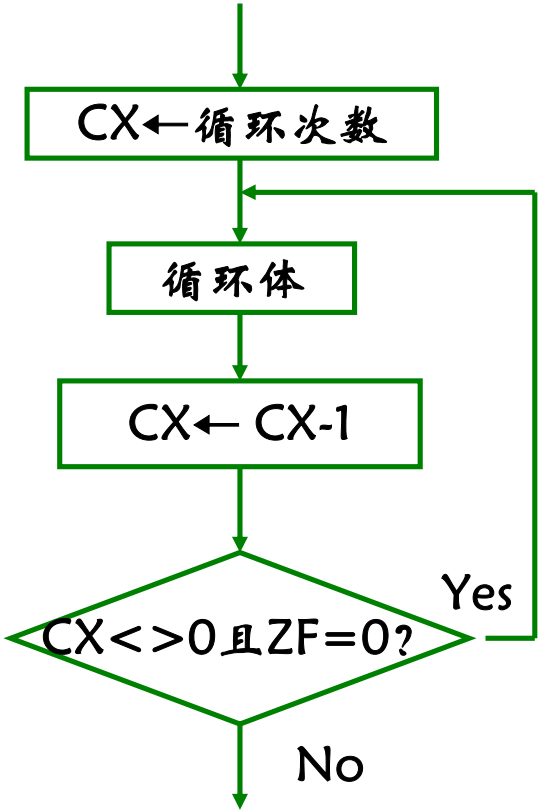
控制转移指令

不相等/不为零循环指令LOOPNE/LOOPNZ

语句格式：LOOPNE/LOOPNZ 短标号

执行过程：

1. (CX)＝(CX)-1（不改变任何标志位）
2. 如果CX≠0且ZF=0，则程序转到循环体的第一条指令；否则，程序将执行该循环指令下面的指令。



控制转移指令

例：记录附加段一个长度为count的字符串中的空格个数到RESULT单元。

```
        MOV   CX，COUNT          ；设置循环次数
        MOV   SI，OFFSET STRING
        XOR   BX，BX             ；BX清0，用于记录空格数
        MOV   AL，20H            ；空格的ASC码为20H
AGAIN:  CMP   AL，ES:[SI]
        JNZ   NEXT               ；ZF=0，非空格，转移
        INC   BX                 ；ZF=1，是空格，个数加1
NEXT:   INC   SI
        LOOP  AGAIN              ；字符个数减1，不为0继续循环
        MOV   RESULT，BX         ；保存结果
```