



普通中等专业教育机电类规划教材

微型计算机接口技术 及应用实习指导

湖南省机械工业学校 廖哲智 主编



0364-43
L22/1

社

机械工业出版社

7p364-43
LZZ/1

普通中等专业教育机电类规划教材

微型计算机接口技术及应用 实习指导

主编 廖哲智

参编 王世峰 李学东 彭曙蓉

主审 徐仁贵

机械工业出版社

052468

本书是根据原机械工业部中专计算机专业教学指导委员会审定的《微型计算机接口技术及应用实习指导》教学大纲进行编写的,为机械工业部“八五”规划教材《微型计算机接口技术及应用》的实习配套教材。

全书共分四部分 25 个实习课题,涵盖了教材的全部内容,并在第三部分安排了电子钟、步进电动机控制、顺序控制、温度巡回检测、汉字显示屏、LCD 显示器等单片机在检测与过程控制及智能仪表方面的应用实例。第四部分介绍了 DVCC-5298H 开发系统的组成、性能和使用方法。并附有 MCS-51 指令表和常用接口芯片引脚图。

本书的课题有难有易,内容循序渐进,便于自学。考虑到目前单片机开发装置系列较多,所以本书各课题实习不涉及具体的实习装置。

本书为中专计算机应用专业教材,也可作为其它相近专业及成人教育教材,还可供工程技术人员和微机爱好者参考。

微型计算机接口技术及应用 实习指导

湖南省机械工业学校 廖哲智 主编

责任编辑:王小东 版式设计:冉晓华

封面设计:海之帆 责任校对:韩 晶

责任印制:王国光

*

机械工业出版社出版(北京市百万庄大街 22 号)

邮政编码:100037

(北京市书刊出版业营业许可证出字第 117 号)

北京第二外国语学院印刷厂印刷

新华书店北京发行所发行·新华书店经售

*

开本 787×1092 1/16·印张 7.5 ·字数 176 千字

1998 年 12 月第 1 版 第 1 次印刷

印数 0001-5000 定价:10.00 元

*

ISBN 7-111-06791-6/TP·923

凡购本书,如有缺页、倒页、脱页,由本社发行部调换

JS402/21

前 言

本书是根据原机械工业部“九五”教材编审出版计划,按照原机械部中专计算机专业教学指导委员会审定的《微型计算机接口技术及应用实习指导》教学大纲进行编写的,由该教学指导委员会审定并推荐出版,为机械工业出版社已出版的“八五”规划教材《微型计算机接口技术及应用》的实习配套教材。

全书共分四部分 25 个实习课题。第一部分为程序设计与调试,综合了顺序、分支、循环和子程序四种基本结构的编程技术,同时又包括了位操作和控制转移及查表等在单片机应用中最经常使用的场合所需要的编程技术;第二部分为硬件及接口技术,紧扣教材内容,依次给出了存储器扩展与 EPROM 固化,基本 I/O 接口和中断技术;8255、8155、8253 应用;定时器/计数器、串行通信、数码显示、键控编程、微型打印机、开关量输入/输出、A/D 及 D/A 转换等 13 个实习课题,涵盖了微机接口技术的全部内容。第三部分为综合应用,安排了电子钟、步进电动机控制、顺序控制、温度巡回检测、汉字显示屏、LCD 显示器等单片机在检测与过程控制及智能仪表方面的应用实例。第四部分为附录,简要介绍了江苏启东计算机厂生产的 DVCC-5298H 开发系统的组成、性能和使用方法。并附有 MCS-51 指令表和常用接口芯片引脚图。

本书在课题选择上有难有易,可供不同专业学生选用;在内容安排上,循序渐进,便于自学;在课题实施方面,考虑到目前单片机开发装置系列较多,所以本书各课题实习均不涉及具体的实习装置。因此这组实习课题对学生的训练是全面的,对于工程技术人员也可以结合本书内容进行学习,从中得到收益和提高,协助解决其所遇到的实际问题。

本书由湖南省机械工业学校廖哲智主编,湖南省机械工业学校王世峰、山东省机械工业学校李学东和湖南省供销学校彭曙蓉参编,山东省机械工业学校徐仁贵主审。

其中,廖哲智编写课题一、二、三、八、十七。王世峰编写课题四、五、六、十三、十四、十五、二十、二十二、二十四。李学东编写课题七、九、十二、十六、十八、十九、二十一、二十三、二十五。彭曙蓉编写课题十、十一。全书由王世峰协助修改定稿。

参加审稿会议的有河北省机电学校曹振军、中原机械工业学校王伟、北京仪器仪表工业学校蒋湘若、陶砂,芜湖机械学校周光辉、江西省机械工业学校李旺达、山东省机械工业学校岳斌、咸阳机器制造学校董少明。他们对本书提出了许多宝贵意见。湖南省机械工业学校陈永革、李佳绘制了本书所有图稿,在此一并表示诚挚的谢意。

限于编者水平,对于本书存在的问题,敬请广大读者批评指正。

编 者

1998 年 7 月

目 录

前言	
第一部分 程序设计与调试	1
课题一 数据传送	1
课题二 数据运算	5
课题三 分支程序	7
课题四 位操作和控制转移	10
课题五 循环程序	13
课题六 子程序	17
第二部分 硬件及接口技术	20
课题七 存储器扩展与 EPROM 编程	20
课题八 基本 I/O 接口和中断	25
课题九 8255 可编程并行 I/O 接口	29
课题十 8155 可编程多功能接口	32
课题十一 定时/计数器	36
课题十二 8253 可编程计数/定时器	40
课题十三 串行通信	44
课题十四 数码显示	47
课题十五 键控编程	51
课题十六 微型打印机	56
课题十七 开关量的输入/输出	61
课题十八 A/D 转换	65
课题十九 D/A 转换	68
第三部分 综合应用	71
课题二十 电子钟	71
课题二十一 步进电动机控制	75
课题二十二 顺序控制	78
课题二十三 温度巡回检测	82
课题二十四 汉字显示屏	90
课题二十五 LCD 显示器	94
第四部分 附录	97
附录 A DVCC-5298H 单片机开发系统使用简介	97
附录 B 常用接口芯片引脚图	107
附录 C MCS-51 指令系统速查表	111
参考文献	113

第一部分 程序设计与调试

课题一 数据传送

一、实习目的

熟悉 MCS-51 寻址方式，掌握数据传送指令功能及应用。

二、实习内容

(一) 内部 RAM 单元之间的数据传送

1. 将立即数 #33H 送 20H 单元。
2. 将 30H 单元的内容 #67H 送 R1。
3. 将累加器 A 中的内容 #27H 送 R1 指示的内存单元。
4. 将数据 #8AH 从 21H 单元送 20H 单元。

(二) 特殊功能寄存器的数据传送

将立即数 #3052H 送数据指针 DPTR。

(三) 外部 RAM 与累加器之间的数据传送

将 A 累加器的内容 #80H 送外部 RAM 的 2100H 单元。

(四) 程序存储器中数据送累加器 A

将程序存储器 1031H 单元的内容送 A 累加器。

(五) 堆栈操作

1. 将累加器 A 的内容压入堆栈。
2. 接上述操作，将已压入堆栈的内容弹出至 22H 单元。

(六) 数据交换

将累加器 A 的内容与 R4 单元内容交换。

三、实习步骤

(一) 内部 RAM 单元之间的数据传送

1. 用立即寻址方式实现，使用 MOV 20H, #33H 指令。

(1) 查指令表知 MOV 20H, #33H 的机器码是 75 20 33 (注：这种通过查指令表获得机器码的方法称为手工汇编，也可以用机器汇编)。

(2) 将机器码输入到实习装置 (开发系统) 的三个连续单元 (如 2000H, 2001H, 2002H)。

(3) 运行该指令

- 1) 单步运行 输入地址 2000H, 按 STEP 单步运行键。
- 2) 断点运行 设断点于 2003H, 从 2000H 单元开始运行。
- 3) 连续运行 输入地址 2000H, 按 EXEC 运行键。

(4) 检查指令运行结果。(20H) = _____

(5) 练习

- | | |
|-------------------|--------------------|
| 1) MOV A, #4CH | 运行后 (A) = _____ |
| 2) MOV R1, #96H | 运行后 (R1) = _____ |
| 3) MOV @R0, #0B3H | 运行后 ((R0)) = _____ |

2. 用寄存器寻址方式实现, 使用 MOV R1, 30H 指令。

- (1) 查指令表, MOV R1, 30H 翻译成机器码为_____。
- (2) 将该机器码输入到实习装置的两个连续单元 (如 3000H, 3001H)。
- (3) 将数据 #67H 装入 30H 单元。
- (4) 输入起始地址 (3000H) 并从此地址开始单步或断点方式运行该指令。
- (5) 检查累加器 (R1) = _____。

(6) 练习

- | | |
|----------------|--------------------------|
| 1) MOV R5, A | 运行后验证 (R5) 与 (A) 是否相同。 |
| 2) MOV 30H, R4 | 运行后验证 (30H) 与 (R4) 是否相同。 |
| 3) MOV A, R0 | 运行后验证 (A) 与 (R0) 是否相同。 |

3. 用寄存器间接寻址方式实现, 使用 MOV @R1, A 指令。

- (1) 查指令表, MOV @R1, A 翻译成机器码为_____。
- (2) 将该机器码输入到实习装置的某个单元 (如 4000H)。
- (3) 将数据 #27H 装入累加器 A。
- (4) 从 4000H 单元开始单步或断点方式运行该指令。
- (5) 检查累加器 ((R1)) = _____。

(6) 练习

- | | |
|-----------------|----------------------------|
| 1) MOV @R0, 59H | 运行后验证 ((R5)) 与 (59H) 是否相同。 |
| 2) MOV A, @R1 | 运行后验证 (A) 与 ((R1)) 是否相同。 |
| 3) MOV 45H, @R0 | 运行后验证 (45H) 与 ((R0)) 是否相同。 |

4. 用直接寻址方式实现, 使用 MOV 20H, 21H 指令。

- (1) 查指令表, 将 MOV 20H, 21H 翻译成机器码为_____。
- (2) 将该机器码输入到三个连续的地址单元 (如 2000H, 2001H, 2002H)。
- (3) 将数据 #8AH 存入 21H 单元。
- (4) 从 2000H 单元开始单步或断点运行该指令。
- (5) 检查 (20H) = _____。

(6) 练习

- | | |
|---------------|-------------------------|
| 1) MOV A, 71H | 运行后验证 (A) 与 (71H) 是否相同。 |
| 2) MOV 45H, A | 运行后验证 (45H) 与 (A) 是否相同。 |

(二) 特殊功能寄存器的数据传送

使用 MOV DPTR, #3052H 指令。

- (1) 查指令表, 将 MOV DPTR, #3052H 翻译成机器码为_____。
- (2) 将该机器码输入到三个连续的地址单元 (如 2000H, 2001H, 2002H)。
- (3) 从 2000H 单元开始单步或断点运行该指令。

(4) 检查 (DPH) = _____, (DPL) = _____。

(三) 外部 RAM 与累加器 A 间的数据传送

使用 MOVX @DPTR, A 指令。

(1) 查指令表, 将 MOVX @DPTR, A 翻译成机器码为_____。

(2) 将该机器码输入到某地址单元 (如 3000H)。

(3) 将 2100H 装入 DPTR, 数据 # 80H 装入累加器 A。

(4) 从 3000H 单元开始单步或断点运行该指令。

(5) 检查外部 RAM (2100H) = _____。

(6) 练习

1) 设外部 (A) = 21H, 单步或断点运行下列指令

MOV P2, # 10H

MOV R0, # 00H

MOVX @R0, A

验证外部 RAM 中 (1000H) 与 (A) 是否相同。

2) 设 (DPTR) = 2200H 单元, 单步或断点运行 MOVX A, @DPTR 指令后验证 (A) 与外部 RAM 中 (2200H) 是否相同。

(四) 程序存储器中数据送累加器 A 的指令

使用 MOVC A, @A + DPTR 指令。

(1) 查指令表, 将 MOVC A, @A + DPTR 翻译成机器码为_____。

(2) 将该机器码输入到某地址单元 (如 3000H)。

(3) 将数据 # 31H 存入累加器 A, # 1000H 存入指针 DPTR。

(4) 从 3000H 单元开始单步或断点运行该指令。

(5) 检查 (A) 与程序存储器中 (1031H) 是否相同。

(6) 练习

设 (A) = 10H, 程序存储器中 (2010H) = 58H, 执行指令 MOVC A, @A + PC (该指令机器码存放在 2000H 单元) 后, (A) = _____。

(五) 栈操作

(1) 压入 (入栈) 操作 使用 PUSH ACC 指令。

1) 查指令表, 将 PUSH ACC 翻译成机器码为_____ (注: 累加器 A 的直接地址为 E0H)。

2) 将该机器码输入到某地址单元 (如 3000H)。

3) 将数据 # 10H 存入堆栈指针 SP, # 83H 存入累加器 A。

4) 从 3000H 单元开始单步或断点运行该指令。

5) 检查 (SP) = _____, ((SP)) = _____。

(2) 弹出 (出栈) 操作 承上述压入操作, 接着使用 POP 22H 指令。

1) 查指令表, 将 POP 22H 翻译成机器码为_____。

2) 将该机器码输入到某地址单元 (如 4000H)。

3) 从 4000H 单元开始单步或断点运行该指令。

4) 检查 (SP) = _____, (A) = _____, (22H) = _____。

(六) 数据交换指令

使用 XCH A, R4 指令。

(1) 查指令表, 将 XCH A, R4 翻译成机器码为_____。

(2) 将该机器码输入到某地址单元 (如 2000H)。

(3) 将数据 #97H 存入累加器 A, 数据 #0B6H 存入 R4。

(4) 从 2000H 单元开始单步或断点运行该指令。

(5) 检查 (A) = _____, (R4) = _____。

(6) 练习

1) XCH A, 74H

运行后验证 (A) 与 (74H) 数据交换情况

2) XCHD A, @R0

运行后验证 (A) 与 ((R0)) 数据交换情况

四、预习要求

1. 熟悉单片机开发装置的使用。
2. 学会手工汇编的方法。
3. 将练习内容手工汇编成机器码。

五、实习报告内容

1. 总结指令的汇编和调试方法。
2. 整理练习中已调试验证的指令, 并做好实习记录。
3. 回答问题

(1) 什么是计算机的指令和指令系统?

(2) 为什么要把汇编语言程序翻译成机器语言程序才能送入计算机运行?

课题二 数据运算

一、实习目的

熟悉数据运算指令功能，学习程序设计的基本方法与调试步骤。

二、实习内容

(一) 双字节十进制无符号数加法

将 31H、30H 单元与 41H、40H 单元的双字节十进制无符号数相加，结果存入 32H、31H、30H 单元。即 (31H) (30H) + (41H) (40H) → 32H、31H、30H

(二) 拆字实习

将片内 RAM II 单元 (如 20H) 的内容拆为两段，并将它们分别存于 II 与 II + 1 单元中 (高位清零)。

(三) 逻辑运算

编程实现下列逻辑运算 $Z = (M \oplus N) \cdot (O + P)$ 。设 M、N、O、P 为片内 RAM 30H ~ 33H 单元，输出结果单元 Z 地址为 34H。

三、实习步骤

(一) 双字节十进制无符号数加法

1. 查指令表将下段程序翻译成机器码并输入到实习装置的某块连续单元 (如从 2000H 单元开始的连续单元)。

	参考程序	机器码	检查数据
	MOV R0, # 30H	_____	(R0) = _____
	MOV R1, # 40H	_____	(R1) = _____
	MOV R2, # 02H	_____	(R2) = _____
	CLR C	_____	(PSW.7) = _____
L1:	MOV A, @R0	_____	(A) = _____
	ADDC A, @R1	_____	(A) = _____
	DA A	_____	(A) = _____
	MOV @R0, A	_____	((R0)) = _____
	INC R0	_____	(R0) = _____
	INC R1	_____	(R1) = _____
	DJNZ R2, L1	_____	(R2) = _____
	CLR A	_____	(A) = _____
	MOV ACC.0, C	_____	(A) = _____
	MOV @R0, A	_____	((R0)) = _____

2. 在 31H、30H 单元存入加数如 3018，在 41H、40H 单元存入被加数如 8975。
3. 输入程序首地址 (此处为 2000H) 然后开始单步或断点运行该段程序。
4. 运行过程中检查数据的变化，并在最后检查 (32H) = _____, (31H) = _____, (30H) = _____

= ____。

(二) 拆字实习

1. 将下段程序翻译成机器码并输入到实习装置。

参考程序	机器码	检查数据
MOV R0, #21H	_____	(R0) = _____
MOV A, 20H	_____	(A) = _____
ANL A, #0FH	_____	(A) = _____
MOV @R0, A	_____	((R0)) = _____
INC R0	_____	(R0) = _____
MOV A, 20H	_____	(A) = _____
SWAP A	_____	(A) = _____
ANL A, #0FH	_____	(A) = _____
MOV @R0, A	_____	((R0)) = _____

2. 在某单元 (如 20H) 存入待拆字的某数 (如 5BH)。

3. 单步或带断点运行该段程序。

4. 运行过程中注意检查数据变化, 并在最后检查 (21H) = _____, (22H) = _____

(三) 逻辑运算

1. 将下段程序翻译成机器码并输入到实习装置。

参考程序	机器码	检查数据
START: MOV A, M	_____	(A) = _____
XRL A, N	_____	(A) = _____
MOV B, A	_____	
MOV A, O	_____	(A) = _____
ORL A, P	_____	(A) = _____
ANL A, B	_____	(A) = _____
MOV Z, A	_____	(Z) = _____

2. 在 M ~ P (30H ~ 33H 单元) 装入待进行逻辑运算的数据 (自设)。

3. 单步或带断点运行该段程序。

4. 运行过程中注意数据的变化情况, 并在最后验证 (Z) 是否与理论值相符。

四、预习要求

1. 看懂实习步骤中所列程序, 注意源程序与目标程序的地址对应, 并预计每一步执行的结果。

2. 编写程序实现逻辑运算: $Z = (X \cdot Y) \oplus Z$

3. 编写 16 位二进制数求补程序。

五、实习报告内容

1. 整理程序和运行结果, 并做好记录。

2. 总结数据运算程序的编写和调试方法。

课题三 分支程序

一、实习目的

掌握分支程序设计及散转程序的设计和调试方法。

二、实习内容

(一) 分支程序

比较存放在 ONE、TWO 和 TRE 三个单元的无符号数，并将小数存放在 20H 单元中。

(二) 散转程序

根据 Rn 单元内容 (00H ~ 7FH)，分别转向程序 ROUT00 ~ ROUT7F (用转移指令表或转移地址表两种方法实现散转)。

三、实习步骤

(一) 分支程序调试

1. 输入程序。
2. 在 ONE、TWO 和 TRE 三个内部 RAM 单元内装入待比较的无符号数。
3. 单步运行，注意 PC 变化情况，同时检查 Cy 状态。
4. 检查 20H 单元的内容，验证是否为最小值。
5. 修改 ONE、TWO 和 TRE 单元置数，重新运行。

(二) 散转程序调试

1. 输入程序。
2. 为便于实习，可缩小散转范围，运行前在 Rn 中装入的数限定为 00H ~ 03H 中的某一个。

3. 确定 JMPTAB 及 PROG00 ~ PROG03 的地址，并在 PROGn 处写入处理程序 (调用系统程序或自编具有针对性的简单语句)。

4. 单步或断点运行，观察并记录 A、DPTR 以及 PC 变化情况。
5. 修改 Rn 中数据，重做。

四、预习要求

1. 复习比较转移指令以及分支程序的编写方法。
2. 复习散转指令以及实现散转的不同方法。
3. 编写并汇编能实现实习内容要求的程序，拟好实习记录表格。
4. 思考两种散转各有什么特点？

五、实习报告内容

1. 整理已通过的实习程序。
 2. 整理实习记录 (包括中间过程和运行结果)。
 3. 回答问题
- (1) 说明分支与散转的区别及编程方法。

(2) 编写并调试符号函数 $y = \text{SGN}(x)$ 的运算程序。 $y = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$

(3) 如果要在三个数中找出最大数或者中间数，应当如何编程？试画出流程图。

六、实习程序参考框图

分支程序参考框图如图 1-3-1 所示，散转程序参考框图如图 1-3-2 所示。

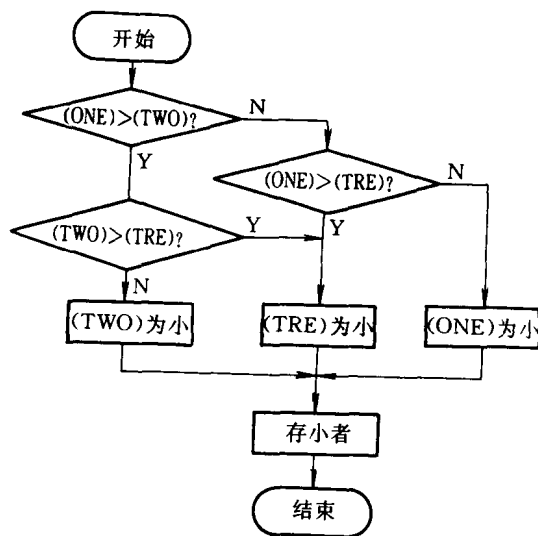


图 1-3-1 比较三数大小的分支程序参考框图

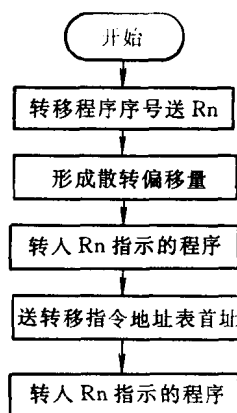


图 1-3-2 散转程序参考框图

七、参考程序

(一) 分支程序

```

MOV  A, ONE
CJNE A, TWO, RR      ; 比较 (ONE) 和 (TWO)
RR:  JC  SS
MOV  A, TWO
CJNE A, TRE, TT      ; 比较 (TWO) 和 (TRE)
TT:  JC  WW
     SJMP UU
SS:  CJNE A, TRE, VV   ; 比较 (ONE) 和 (TRE)
VV:  JC  WW
UU:  MOV  A, TRE
WW:  MOV  20H, A       ; 存小者
  
```

(二) 散转程序

1. 用转移指令表实现散转

```

MOV  A, Rn
RL   A               ; 形成散转偏移量
MOV  DPTR, #JMPTAB  ; 转移指令表首址
  
```

```

        JMP    @A + DPTR
JMPTAB: AJMP  PROG00                ; 转移指令表
        AJMP  PROG01
        :
        AJMP  PROGnn
PROG00: MOV   20H, # 00H            ; 转入 Rn 指示的程序
        SJMP  $
        :
PROGnn: MOV   20H, # nH
        SJMP  $
2. 用转移地址表实现散转
        MOV   DPTR, # TAB          ; 转移地址表首址
        MOV   A, Rn
        ADD   A, Rn                ; 形成散转偏移量
        JNC   LOOP
        INC   DPH
LOOP:   MOV   Rm, A                ; 查出偏移地址的高 8 位
        MOVC  A, @A + DPTR
        XCH   A, Rm
        INC   A
        MOVC  A, @A + DPTR        ; 查出偏移地址的低 8 位
        MOV   DPL, A
        MOV   DPH, R3
        CLR   A
        JMP   @A + DPTR          ; 转入 Rn 指示的程序
TAB:    PROG00, PROG01, ..., PROG nn
PROG00: MOV   20H, # 00H          ; 转入 Rn 指示的程序
        SJMP  $
PROG01:
        :
PROGnn: MOV   20H, # nnH
        SJMP  $

```

课题四 位操作和控制转移

一、实习目的

熟悉位操作指令和控制转移指令功能。

二、实习内容

(一) 位操作

编程实现逻辑运算： $Q = (U \cdot (V + W)) + (X \cdot \bar{Y}) + \bar{Z}$ 。设 $U \sim Z$ 的位地址为 $00H \sim 05H$ ， Q 的位地址为 $07H$ 。（实际中 $U \sim Z$ 可以是外部输入端口信号或软件设定的一些控制位）。

(二) 控制转移

编程实现字符串长度统计：设在单片机片内 RAM 中从 STR 单元开始有一字符串（以 ASCII 码存放），该字符串以 \$（其值为 $24H$ ）结束，试统计该字符串的长度，其结果存于 LON 单元。

三、实习步骤

(一) 位操作

1. 输入所编程序。
2. 在内部 RAM $20H$ 单元（位地址是 $00H \sim 07H$ 的位在 $20H$ 单元）输入某个数据（注意此时各位的状态并做好记录）。
3. 单步或断点运行所编程序，逐步检查 C_y 及各数据位的变化情况（注意检查时也是按字节读出，再观察其相应位）。
4. 验证运行结果。
5. 改变 $20H$ 单元的置数，重做。

(二) 控制转移

1. 输入所编程序。
2. 在以 STR 为首地址的内部 RAM 单元开始存放待统计长度的数据块（自设），并以 $\#24H$ 作为数据块的结束标志。
3. 单步或断点运行所编程序，注意观察每次转移后 PC 及统计长度单元 LON 的变化，并作好记录。
4. 验证运行结果。
5. 改变数据块长度，重做。

四、预习要求

1. 阅读教材，熟悉位操作指令和控制转移指令的功能和特点。
2. 按照实习内容编写并汇编实习程序，并拟好记录表格。
3. 确定待进行位操作运算的数据，并计算出结果以验证所编程序。
4. 确定待统计长度的字符串首地址和其长度以及长度存放单元的地址。
5. 掌握无条件转移、条件转移的作用和运用特点。

五、实习报告内容

1. 整理所编程序和实验记录。
2. 说明位操作和字节操作有何区别？
3. 条件转移中的绝对转移和相对转移各有何特点，使用上又有什么区别？
4. 试编程并调试位操作程序 $Q = (J + \overline{K}) \cdot L$

六、实习程序参考框图

位操作实习参考框图如图 1-4-1 所示。控制转移实习（数据块长度统计）参考框图如图 1-4-2 所示。

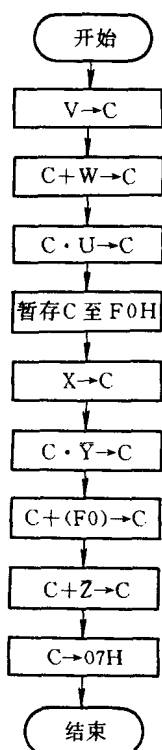


图 1-4-1 位操作实习程序参考框图

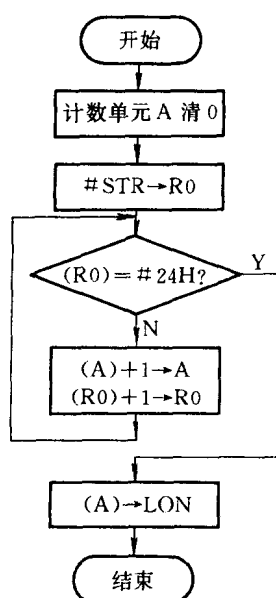


图 1-4-2 控制转移实习程序
(数据块长度统计) 参考框图

七、参考程序

(一) 位操作

```

MOV  C, 01H
ORL  C, 02H
ANL  C, 00H
MOV  F0, C
MOV  C, 03H
ANL  C, /04H
ORL  C, F0
ORL  C, /05H
MOV  07H, C
  
```

；暂存 $U \cdot (V + W)$ 的值

；保存结果

(二) 控制转移

```
START: CLR  A                      ; 计数单元清零
        MOV  R0, # STR
LOOP:   CJNE @R0, # 24H, NEXT       ; 判断字符串是否结束
        SJMP COMP
NEXT:   INC  A
        INC  R0
        SJMP LOOP
COMP:   MOV  LON, A                 ; 将长度存入 LON 单元
        SJMP $
```