



[12]发明专利申请公开说明书

[21]申请号 94118754.3

[51]Int.Cl⁶

G06F 19 / 00

[43]公开日 1996年7月24日

[22]申请日 94.12.3

[71]申请人 刘若飞

地址 100026北京市朝阳区马道口东街甜水园
东里44楼2门601

[72]发明人 刘若飞

[74]专利代理机构 北京市西城区专利代理事务所
代理人 许深

权利要求书1页 说明书9页 附图页数10页

[54]发明名称 便携式单CPU仿真器

[57]摘要

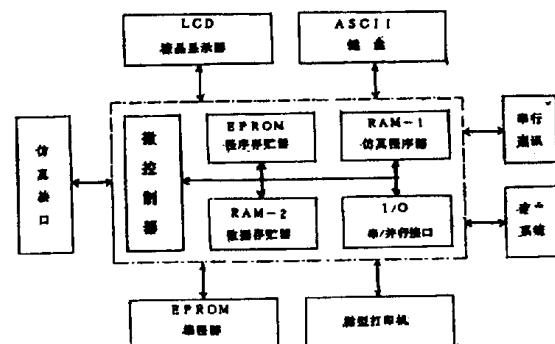
本发明涉及一种单片机仿真器。

单片机仿真器是一种需借助于PC机的软件开发工具，本发明是在系统板上增加了硬件控制电路，使单一CPU仿真器脱离PC机完成监控和仿真功能。

本发明配有标准英文键盘、图形/字符型液晶显示器、微型打印机等，以实现无PC机条件下进行汇编、反汇编、宏汇编程序的输入及调试。

本发明配有语音系统，给出语音提示或读出输入的内容。

本发明通过通讯口连接有用于简化编程的图形库高级语言终端。



权 利 要 求 书

1. 一种便携式单CPU仿真器，其硬件由微控制器、程序存储器、数据存储器、仿真存储器、仿真接口、串/并行接口、EPROM编程器、键盘及显示器组成，其特征在于：

- (1). 还包括有硬件控制电路、语音系统、打印机和通过串行通讯口连接的图形库/高级语言调试终端；
- (2). 键盘是一个标准ASCII键盘；
- (3). 显示器是一个液晶显示器。

2. 根据权利要求1所述的便携式单CPU仿真器，其特征在于液晶显示器可以是字符型的，也可以是图形型的。

3. 根据权利要求1所述的便携式单CPU仿真器，其特征在于打印机是一种微型打印机。

4. 根据权利要求1所述的便携式单CPU仿真器，其特征在于硬件控制电路是由一组或门和触发器组成，由RST信号及地址/数据线信号触发并发出切换信号控制其它硬件电路。

5. 一种便携式单CPU仿真器，其软件包括有机器码输入程序、机器码显示程序、全速运行程序、单步运行程序、断点运行程序、EPROM固化程序、打印程序及通讯程序，其特征在于还包括宏汇编程序、汇编程序、反汇编程序、图形符号输入程序及语音程序。

说 明 书

便携式单CPU仿真器

本发明涉及一种单片微机仿超真器

单片机仿真器是一种软件开发工具，已有技术的单片机仿真器，需借助PC微机才能实现汇编语言及高级语言（PL/M、C语言）调试程序，还需使用昂贵的PC机软件。在脱离PC机后，只能输入和查看16进制的机器码，而且程序的汇编和反汇编工作需要由人工查指令表完成；这些导致了低档仿真器操作复杂繁琐，而高档仿真器对设备的依赖性大，成本高，尤其是对广大的青少年的计算机爱好者难以普及。

本发明的目的在于提供一种完全摆脱对PC机的依赖，配置一些简单的外设，以单片机汇编语言实现宏汇编和高级语言调试单片机程序的技术，填补了中档仿真器的空白。

本发明由硬件和软件两部分组成

硬件由基本系统及扩展系统组成，基本系统由系统板、显示器及键盘组成，系统板由单片微控制器（CPU）、程序存储器、数据存储器、仿真存储器及串/并行扩展接口组成，扩展系统有EPROM编程器，其特征是：

基本系统中：

- 1、系统板上还有：仿真信号控制电路、硬件控制电路及掉电保护电路；
- 2、键盘是标准的ASCII键盘；
- 3、显示器是LED图形或字符型液晶显示器。

扩展系统中：

- 1、通过I/O口接有微型打印机；
- 2、通过I/O口还接有语音系统；
- 3、通过RS-232接口可接有高级语言调试终端或图形库终端。

软件由监控程序控制调用各有关子程序，如机器码输入输出程序、机器码显示程序、全速运行程序、断点运行程序、单步运行程序、EPROM固化程序和串行通讯程序，其特征是：还有汇编程序、反汇编程序、宏汇编程序、图形符号输入程序及语音程序。

微控制器的数据/地址总线分别与存贮器、控制电路、并行扩展接口和显示器相接，其控制信号与硬件控制电路相接，分别再控制其它电路，其P₀、P₁、P₂及P₃均与仿真接口相接，作为仿真信号。本发明的微控制器既完成程序的输入输出和调试工作，又实现单片机仿真功能，实现了单一CPU型的仿真器。

程序存贮器为一EPROM，固化有整个系统的监控程序，其数据/地址线与微控制器相接，片选、读信号由硬件控制电路发出。

仿真存贮器为静态存贮器，其数据/地址线与微控制器相接，片选、读写信号由硬件控制电路发出，内装有用户输入的仿真程序。

数据存贮器为静态存贮器，其数据/地址线与微控制器相接，片选、读写信号由硬件控制电路发出，内装有用户输入的源程序，图形符号代码及微控制器的状态信息。

硬件控制电路和仿真信号控制电路将微控制器发出的控制信号经缓冲器缓冲后送到硬件控制电路，再分别控制仿真信号控制电路和其它电路。为使单一微控制器既能完成程序开发，又能仿真出让，需实现两种状态的转换，本发明是采用“数据地址信号转让法”，硬件控制电路是一组或门和触发器组成，当数据线和地址线上均为“1”时，触发器被触发，并发出切换信号以控制信号转换（或通向本系统板或通向仿真接口），当仿真器复位时，触发器为初始状态（监控系统状态）。

掉电保护电路作为仿真存贮器和数据存贮器的掉电保护之用。

程序的汇编语言输入和输出：使用者可通过本发明连接的标准 ASCII 键盘输入汇编语言、指令，仿真器对当前的语言进行汇编（无需事先进行人工汇编）并存入指定的地址中，整个用户界面模拟 PC 机的 DEBUG 系统，可不占用大量的内存存贮汇编语言（不具有“带符号的地址跳转”、“伪指令”及“模块链接”等宏汇编功能）仿真器在反汇编程序时，通过液晶显示器输出汇编语句，还可由微型打印机打出。

程序的宏汇编语言及图形符号输入：使用者在汇编语言输入的基础上，使用“带符号的地址跳转”、“伪指令”及“模块链接”等宏汇编功能，本发明的仿真器将输入的汇编语句进行压缩（省略过多的空格符）并存入数据存贮器，待输入完毕后进行宏汇编（无需对偏移地址进行计算）以便于在程序的输入过程随时进行修改。图形符号输入程序是在宏汇编的基础上，添加“图形符号汇编语言”程序，图形符号所对应的语句、子程序，均事先固化在 EPROM 中并制出图形符号表相对照，用户按图形符号输入表中所指定的地址读出并存贮在数据存贮器中，当系统对其编译时，是调用宏汇编对有关的语句、子程序进行宏汇编生成机器码，其中所有的图形符号均存放于图形库终端中。

程序的语音读出及语音提示：当对程序进行反汇编后，可通过附加的语音系统将其逐条读出，使用者可不必看显示器便知道存贮器中的程序，适于特殊环境下使用。对本发明的仿真器进行操作时出现的错误亦可通过语音系统读出并改正之。

程序的高级语言调试：由于单 CPU（监控、仿真共用 CPU）本身功能的限制，为能完成如“C、PL/M”等高级语言的交叉汇编调试，可在本发明仿真器的串行通讯接口附加一高级语言调试终端（自带 CPU）即可。

本发明的硬件扩展了外设，内部增加了硬件控制电路，仿真信号控制电路和掉电保护电路。软件增加了宏汇编程序、图形符号输入程序和语音读出程序，使其在无PC机的情况下完成需PC机支持的许多功能。

本发明与已有技术相比，其优点是：

1. 可以脱离PC机独立工作；
2. 可直接用宏汇编编程、调试程序，当配有高级语言或图形库终端时，可进行图形符号和高级语言的编程和调试；
3. 具有语音提示及语音程序助读和错误语音告警功能；
4. 功能齐全，成本低，利于推广，特别适用于教学；
5. 体积小，携带方便。

附图说明：

图1为本发明的系统框图

图2为本发明的硬件电原理图

图3为本发明的监控程序主流程图

图4为本发明宏汇编程序流程图之一

图5为本发明宏汇编程序流程图之二

图6为本发明汇编程序流程图

图7为本发明反汇编程序流程图

图8为本发明语音读出程序流程图

图9为本发明图形符号输入程序流程图之一

图10为本发明图形语言输入程序流程图之二

实施例

本发明的系统框图如图1，由系统板(虚线内)1、液晶显示器2、键盘3、组成基本系统，并配以扩展系统(由EPROM编程器4、微型打

印机5、语音系统6及通过串行通讯接口7接有图形库/高级语言调试终端)。本发明的硬件电原理图如图2。微控制器的数据/地址总线分别与存贮器、控制电路、并行扩展接口和显示器相接，其控制信号与硬件控制电路相接，分别再控制其它电路，其P₀、P₁、P₂及P₃均与仿真接口相接，作为仿真信号。

程序存贮器为一EPROM，固化有整个系统的监控程序，其数据/地址线与微控制器相接，片选、读信号由硬件控制电路发出。

仿真存贮器为静态存贮器，其数据/地址线与微控制器相接，片选、读写信号由硬件控制电路发出，内装有用户输入的仿真程序。

数据存贮器为静态存贮器，其数据/地址线与微控制器相接，片选、读写信号由硬件控制电路发出，内装有用户输入的源程序、图形符号代码及微控制器的状态信息。

硬件控制电路是一组或门和触发器组成，当数据线和地址线上均为“1”时，触发器被触发，并发出切换信号以控制信号转换(或通向本系统板或通向仿真接口)，当仿真器复位时，触发器为初始状态(监控系统状态)。

本发明的软件由监控程序作为主系统进行管理，所有的其它软件的运行均由监控程序调用。本发明除具有常用的仿真器全部运行程序外，还增有宏汇编程序、汇编程序、反汇编程序、图形符号输入程序及语音读出程序。

本发明汇编程序的流程图如图6，其部分源程序如下：

INITIALIZE:	;
MOV DPTR, #CODE_TABLE	初始化
MOV R0, #BUFFER	:将机器码指令表的首地址置于DPTR中
MOV R1, #00H	:将键盘/显示缓冲区首地址置于R0中。
MOV R2, #00H	;
MOV R3, #00H	;
MOV R4, #00H	;
.	;
.	;
.	;

```

MOV R7, #00H          ;内装有汇编的机器码(同时是指令表的指针)
MOV OCH, #00H          ;清内存, 00~0F中分别寄存放语句汇编后的
                        ;机器码入该语句的长度

.
.
.

MOV OFH, #00H          ;
CLR A                 ;清累加器
START:                ;开始
    CJNE @R0, #CDH, REL1 ;看一下第n个字符是否是回车, 即“没有输入”
    MOV A, #01H            ;至“无输入”标志01于A中并返回
    RET
REL1:                 ;将第n个字符先放于B中保存
    MOV B, @R0            ;将机器码中的值读入到A中
    CLR A
    MOVC A, @A+DPTR      ;比较一下是否相同, 若不同则转到NEXT_CODE
    CJNE A, B, NEXT_CODE ;是否是回车, 即无操作数的汇编指令
    CJNE A, #ODH, REL2   ;妆前的R7值就是机器码, R7值同时是指令表的
                        ;指针。将其装入OF中返回 A置零表示汇编成功
    MOV OFH, R7
    CLR A : (成功汇编标志)
    RET
REL2:                 ;是否是空格符, 即有无操作数因为操作数与主
    CJNE A, #20H, REL3   ;机器码间有空格符分开
    INC R0,               ;转到操作数求值程序
    SJMP OPERATION_CODE
REL3:                 ;各指针加一, 继续比较下一个字符
    INC R0,
    INC DPTR
    SJMP REL1
NEXT_CODE:             ;当前比较的指令不同, 换下一个指令继续
    CJNE R7, #0FFH, REL4 ;R7的值到了FF否? 即所有的指令都比较了
    MOV A, #03H            ;否?若是, 置A值为03, 返回
    RET
REL4:                 ;R7值加一, 即指向指令表中下一指令。
    INC R7                ;由R7值算出指令表中下一指令的地址并放入
    MOV A, R7              ;DPTR中
    ANL A, #11110000B      ;高八位地址装入DPTR的高八位上
    SWAP A,               ;低八位地址装入DPTR的低八位上
    ADD A, #TABLE_H
    MOV DPH, A
    MOV A, R7
    ANL A, #00001111B
    SWAP A,
    MOV DPL, A,
    MOV R0, #BUFFER
    LJMP REL1              ;重新置键盘/显示缓冲区首地址
                            ;再进行比较

.
.
.

OPERATION_CODE:       ;比较操作数, 并求值
INITIALIZE:           ;初始化
                        ;(略)

```

反汇编程序的流程图如图7，其部分源程序如下：

```
VN_ASM:  
    MOV A, R7  
    ANL A, #11110000B ;R7中为机器码,将其值放于A中.  
    SWAP A,  
    ADD A, #TABLE_H ;计算出指令表的地址并放于DPTR中  
    MOV DPH, A  
    MOV A, R7  
    ANL A, #00001111B  
    SWAP A  
    MOV DPL, A  
    CLR A  
    MOVO A, @A+DPTR ;读指令表 反汇编  
    CJNE A, #00H, V_1 ;有无操作数,  
    PUSH DPH ;有,将原来用户程序的地址取出,将  
    PUSH DPL ;跟在机器码后的操作数读出反汇编  
    MOV DPH, 7DH  
    MOV DPL, 7EH  
    MOVX A, @DPTR  
    INC DPTR  
    MOV 7DH, DPH  
    MOV 7EH, DPL  
    INC RO  
    LCALL HEX_ASC ;将操作数转为ASCII形式,并放入  
    POP DPL ;显示/键盘缓冲区准备显示.  
    POP DPH  
    INC RO  
    INC DPTR  
    SJMP LOVP ;继续反汇编  
V_1: CJNE A, #0DH, V_2 ;结束否?  
    CLR A ;结束了,置A值为零,成功标志  
    RET  
V_2: MOV @RO, A ;将反汇编出的语句放入显示/键盘  
    INC RO ;缓冲区准备显示  
    INC DPTR  
    SJMP LOOP ;继续反汇编  
  
    . (略)
```

宏汇编程序的流程图如图4及5，其部分源程序如下：

```
CJNE @R0, #3AH, GO-ON      ;是否是“:”符, ASCII码为3AO  
POP R0                      ;因为标号后必须跟“:”符以示与  
LJMP SIGN                   ;程序指令的区别,若是则转到SIGN  
  
GO-ON:  
  
SIGN:  
    MOV DPTR, #TABLE-SIGN   ;处理标号的程序  
    MOV R7, #00H              ;将标号表的首地址放于DPTR中.  
    MOV B, @R0                ;清R7,R7中为标号的表中顺序值.  
    MOV A, @DPTR              ;将第n个字符放入B中  
    CJNE A, B, NEXT-SIGN    ;读标号表中的第n个字符  
    CJNE A, #3AH, REL2       ;是否相同,若不同则转到NEXT-SIGN  
    LCALL ERROR              ;比较完了  
    MOV A, #01H               ;有相同的标号,出错  
    MOV R0, #01H               ;置出错标志01于A中,返回  
    RET  
REL2:  
    INC R0  
    INC DPTR  
    SJMP REL1                ;继续比较  
  
NEXT-SIGN:  
    CJNE R7, #OFFH, REL3     ;向有的标号比较完了?  
    LSJMP GO-ON  
  
REL3:  
    INC R7  
    MOV A, R7  
    ANL A, #11110000B  
    SWAP A  
    ADD A, #TABLE-H  
    MOV DPH, A  
    MOV A, R7  
    ANL A, #00001111B  
    SWAP A  
    MOV DPL, A  
    MOV R0, #BUFFER           ;重新置键盘/显示缓冲区首地址  
    LJMP REL1                ;再进行比较  
  
GO-ON:  
    POP DPH  
    POP DPL  
    LCALL DPTR+              ;取上一次标号表最后一个标号的顺序值  
    MOV R0, #BUFFER           ;并放入DPTR  
    LCALL DPTR+              ;调把DPTR加1b(十进制)的子程序  
    MOV R0, #BUFFER           ;将新标号装入表中,即建立新标号  
  
REL4:  
    MOV A, @R0  
    MOV @DPTR, A  
    CJNE @R0, #3AH, REL5     ;装入完?  
    INC DPTR
```

```

    MOV 0A, 7DH; ;把地址计数器值装入标号表中。
    MOV @DPTR, A
    MOV A, 7EH
    MOV @DPTR, A
    RET; ;返回
    INC R0
REL5: INC R0
    INC DPTR
    SJMP REL4; ;继续装入
    . (略)

```

语音程序的流程图如图8，其部分源程序如下：

```

        LCALL VN_ASM; ;调用反汇编程序
        ANL D1, #11110000B; ;打开语音
        INC R0
        MOV @R0, #ODH; ;在键盘/显示缓冲区的待续字符后置结束
                        ;标志
        MOV R0, #BUFFER; ;置键盘/显示缓冲区的首地址于R0中
LOOP: CJNE @R0, #ODH, REL1; ;全部读完?
        SJMP NEXT_V; ;是的,去读下一条语句
REL1: MOV DPTR, #TABLE2; ;将ASCII 码变控制码的表的首地址置入
                        ;DPTR中
        LOOP1: CLR A
                MOV C A, @A+DPTR; ;读入一个控制码的ASCII码
                MOV B, @R0; ;读入一个待续机器码的ASCII码
                CJNE A, B, REL2; ;相同吗?
                MOV A, DPL; ;DPL的值实际就是控制码的值,将它置于A
                                ;中并调语音读出VOICE.
                LCALL VOICE
                SJMP NEXT_V; ;继续下一个字符的发音
REL2: INC DPTR; ;DPTR加一,读下一个控制码的ASCII值
        SJMP LOOP1; ;继续比较
NEXT_V: INC R0; ;R0指向键盘/显示缓冲区中的下一个字符
        SJMP LOOP; ;继续
        . (略)

```

图形符号输入程序的流程图如图9及10。

示意图

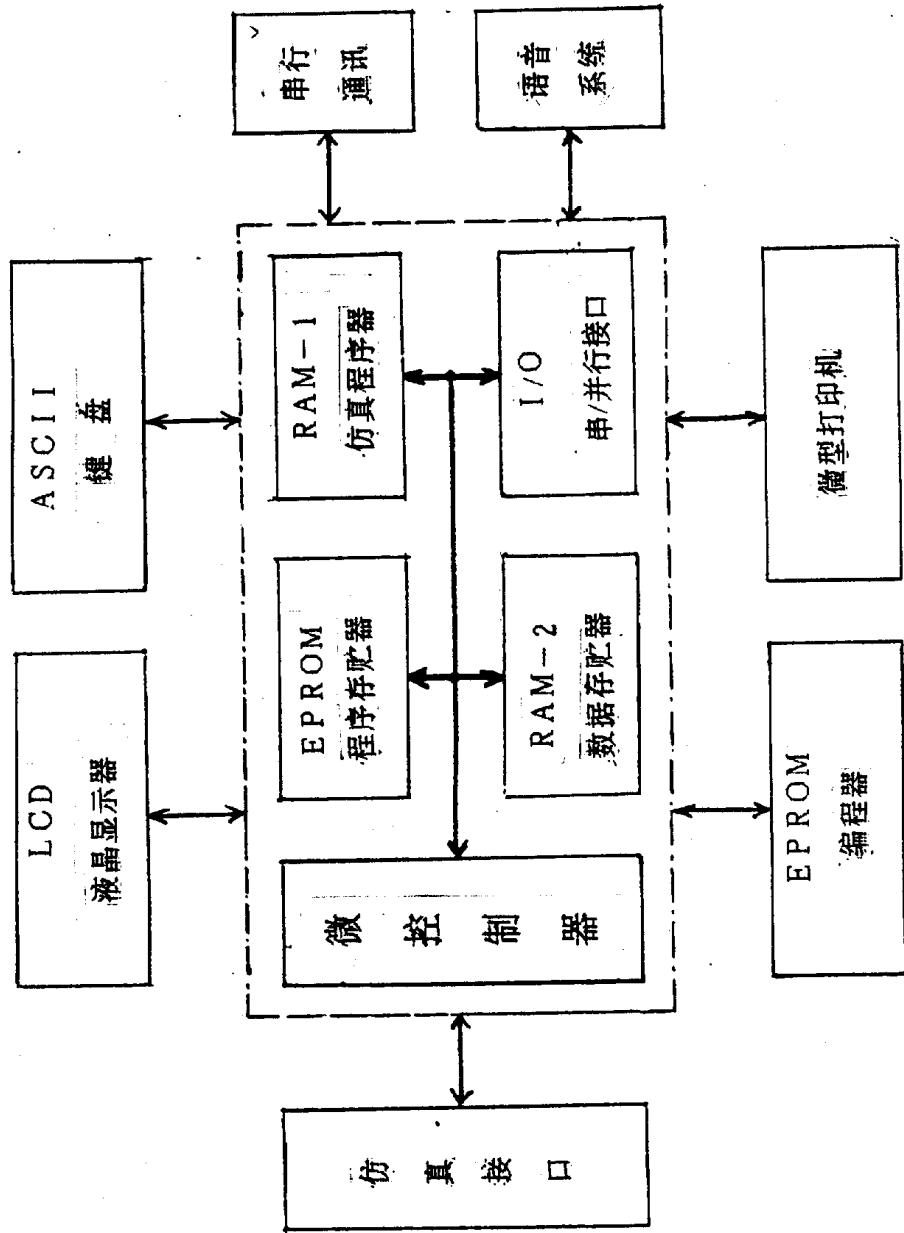
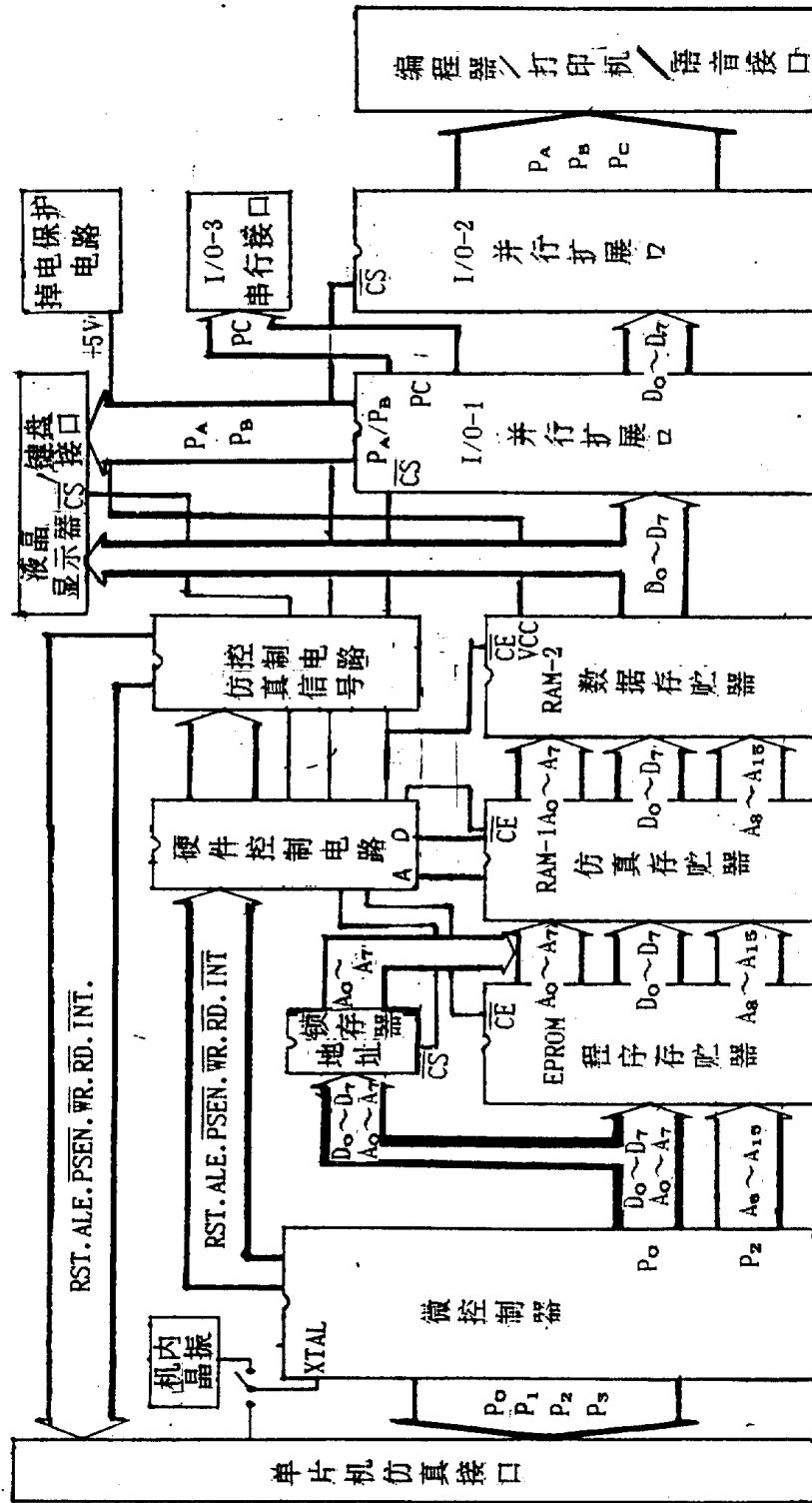


图 1

图 2



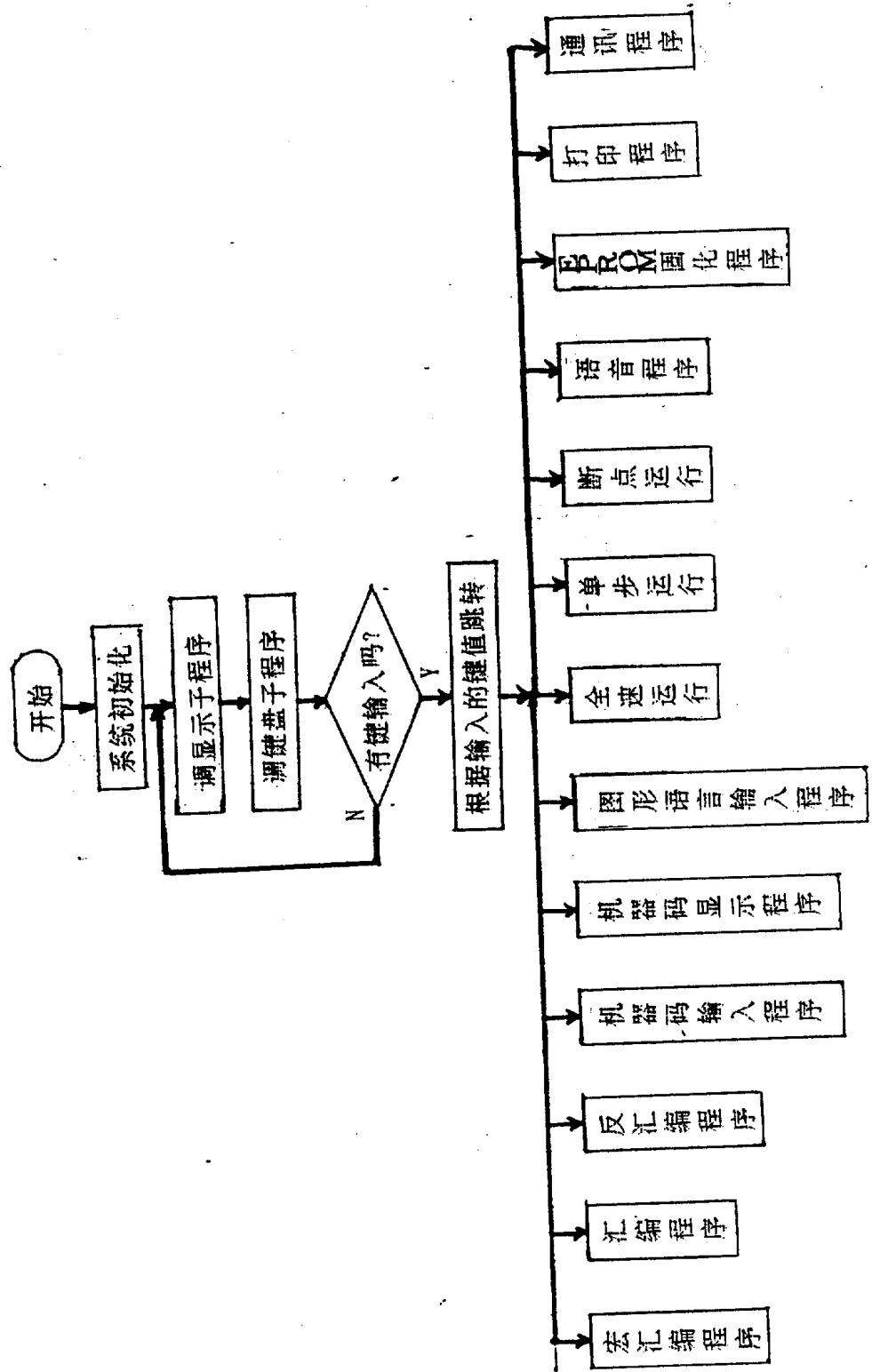


图 3

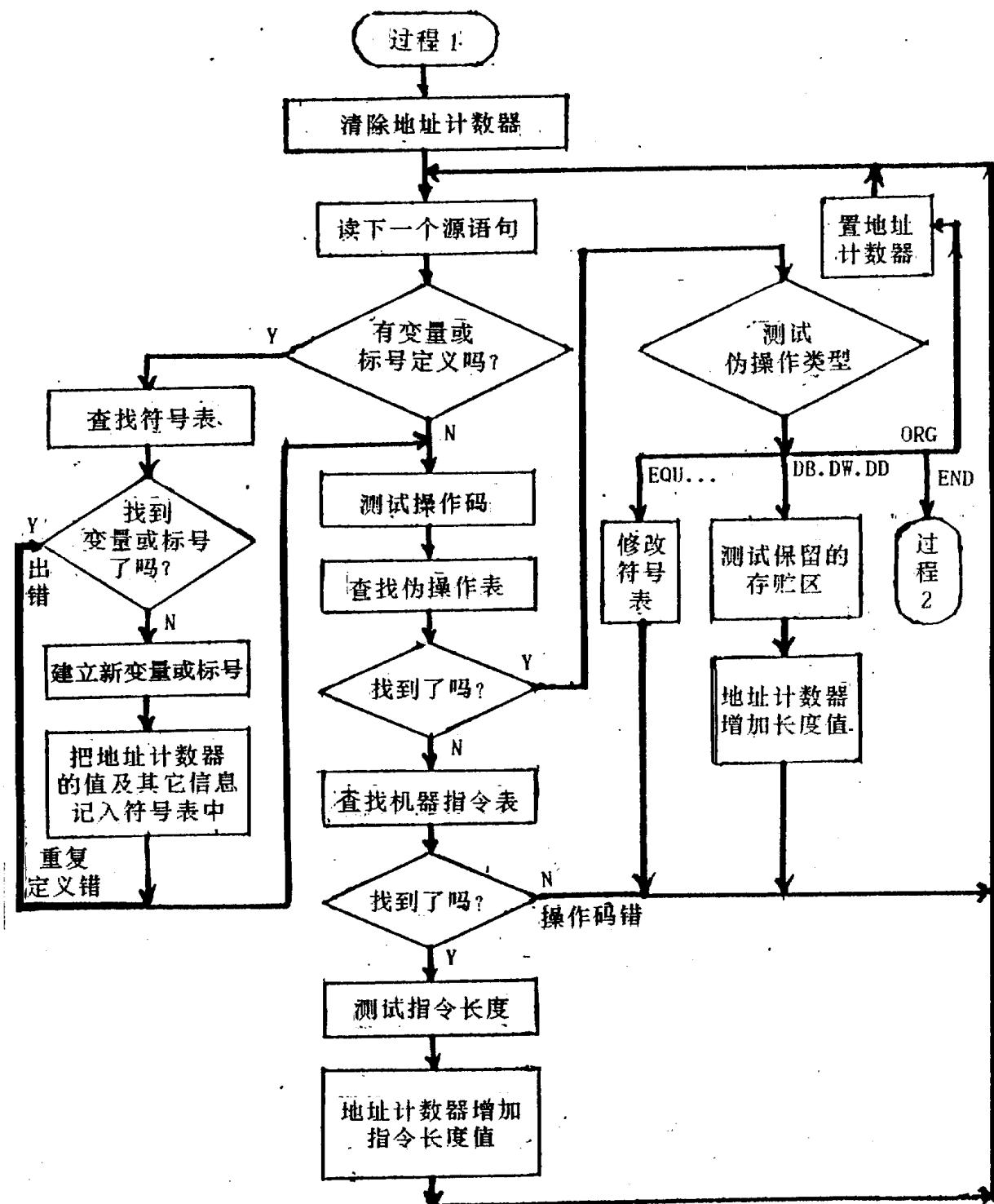


图 4

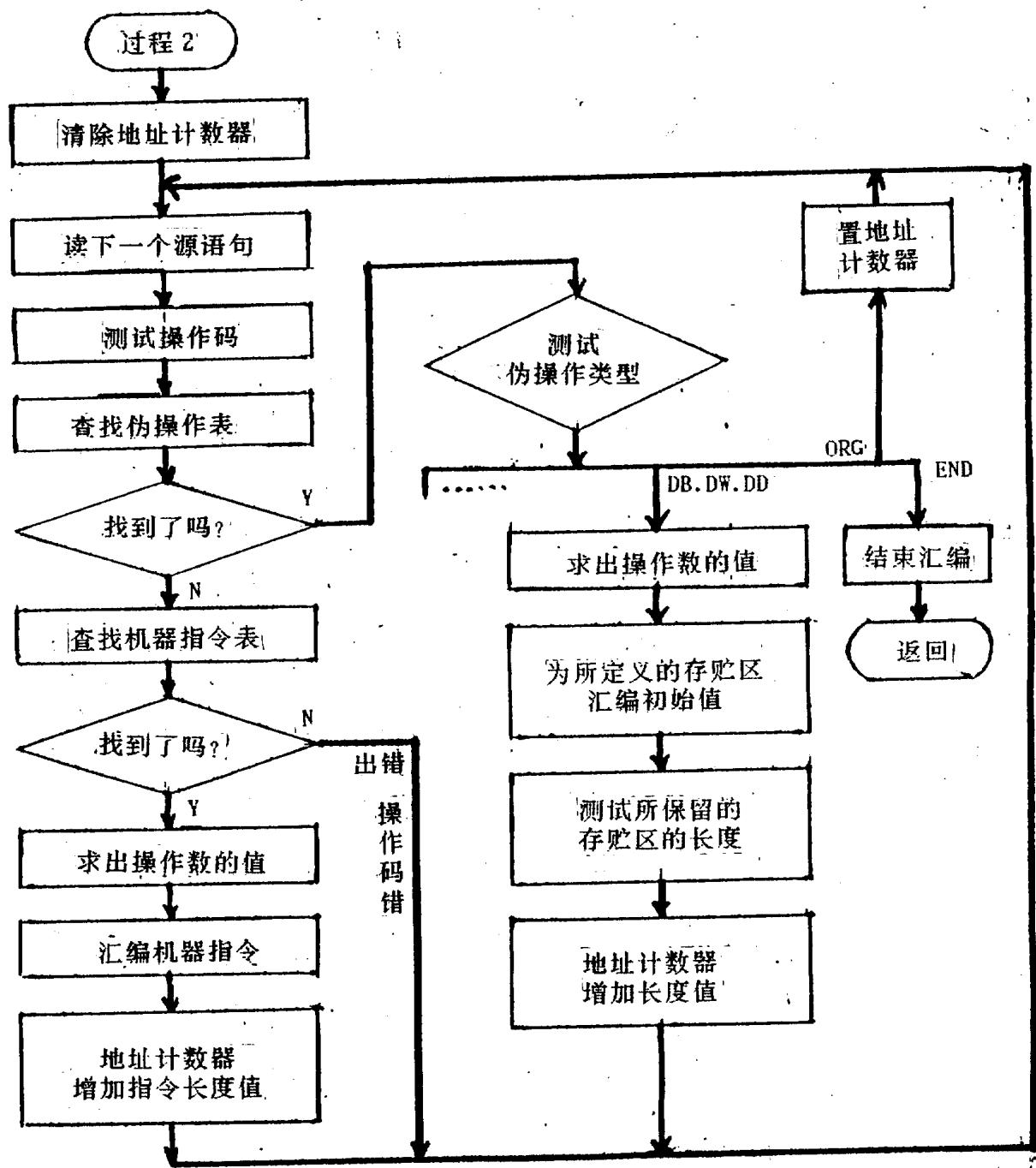


图 5

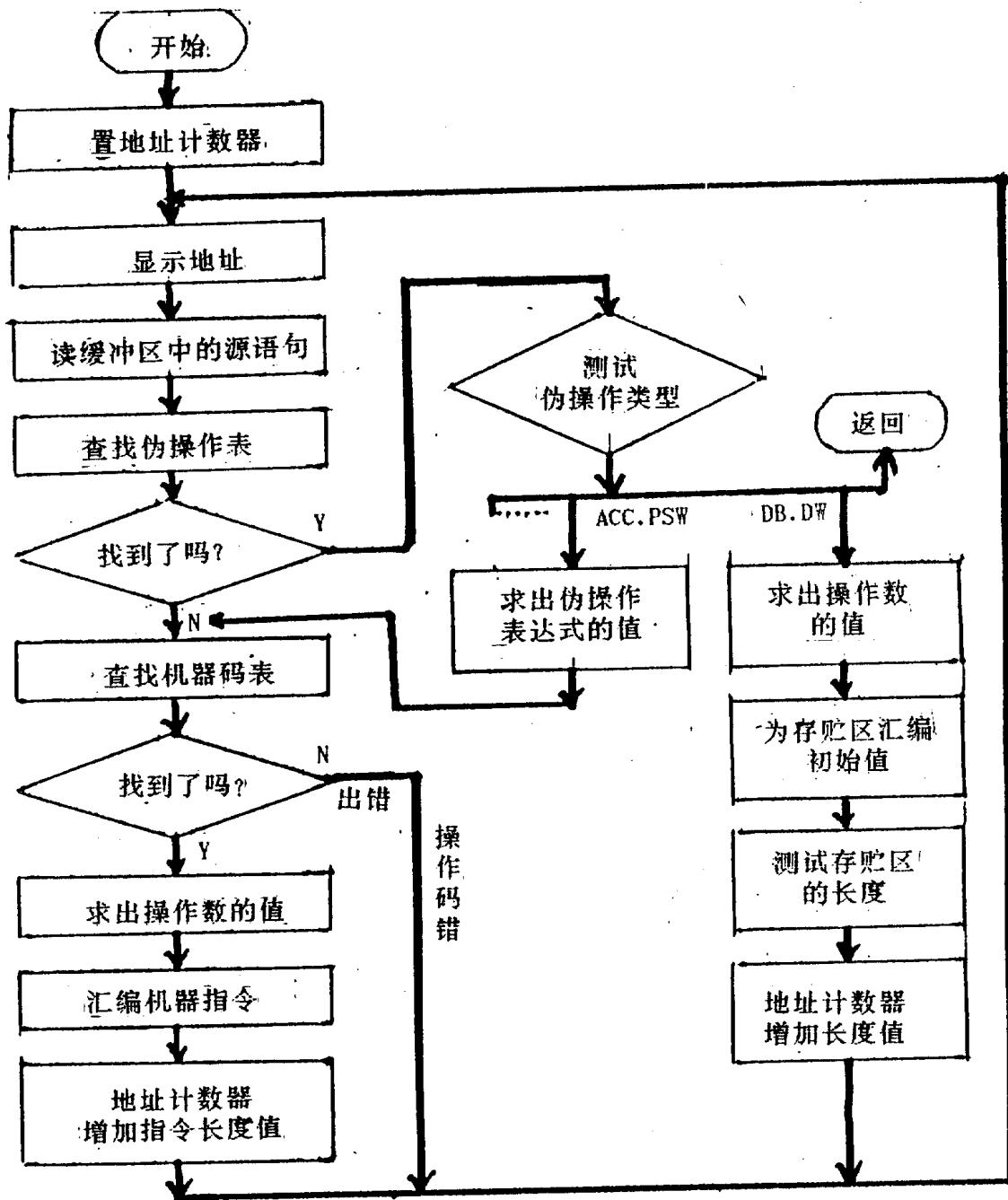


图 6

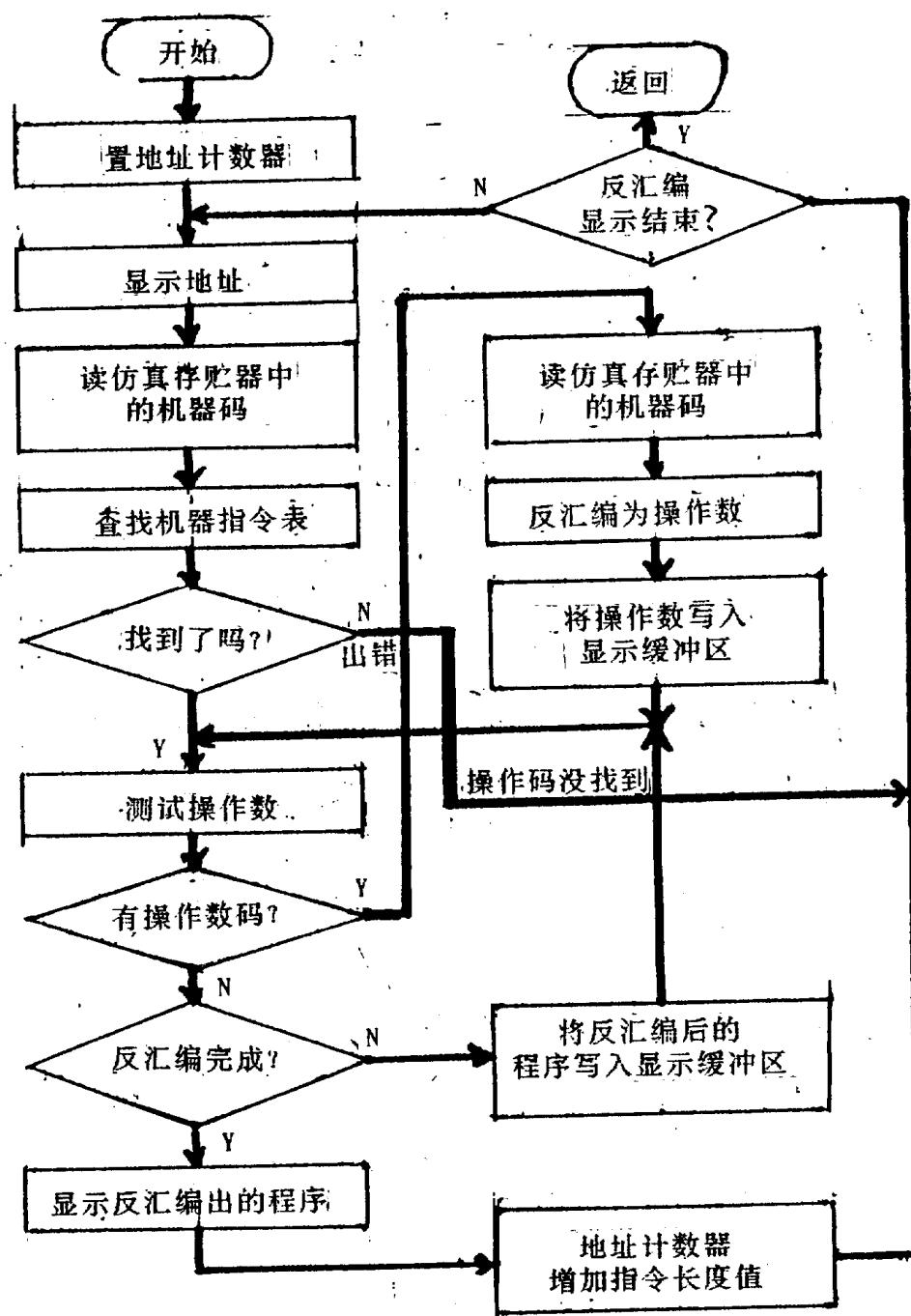


图 7

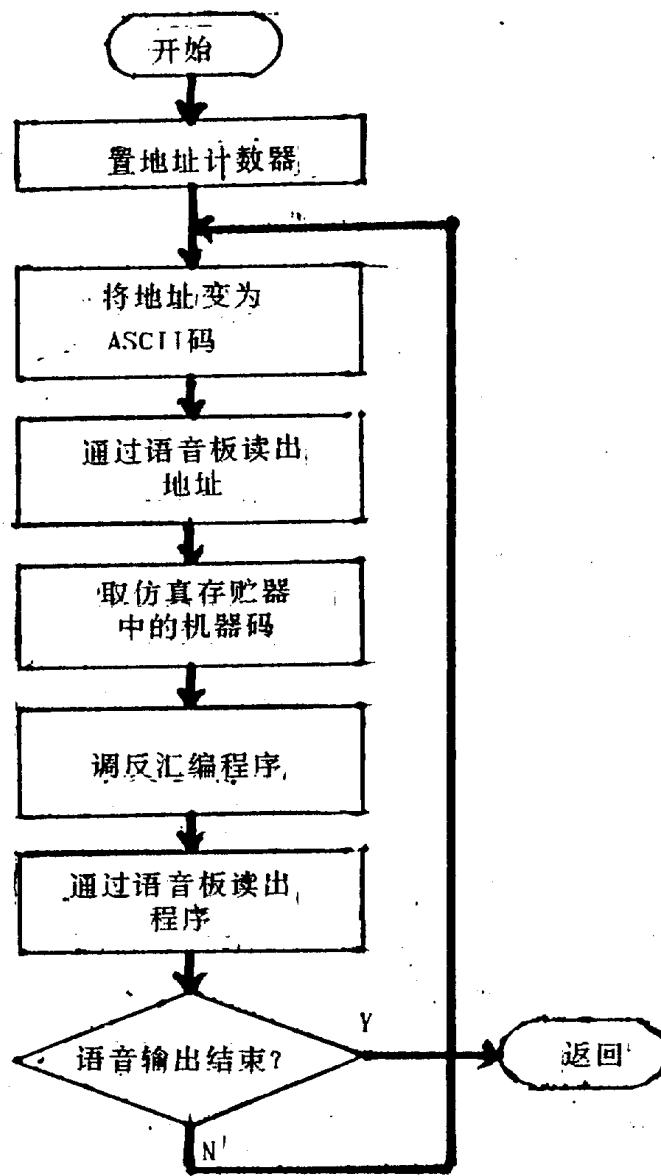


图 8

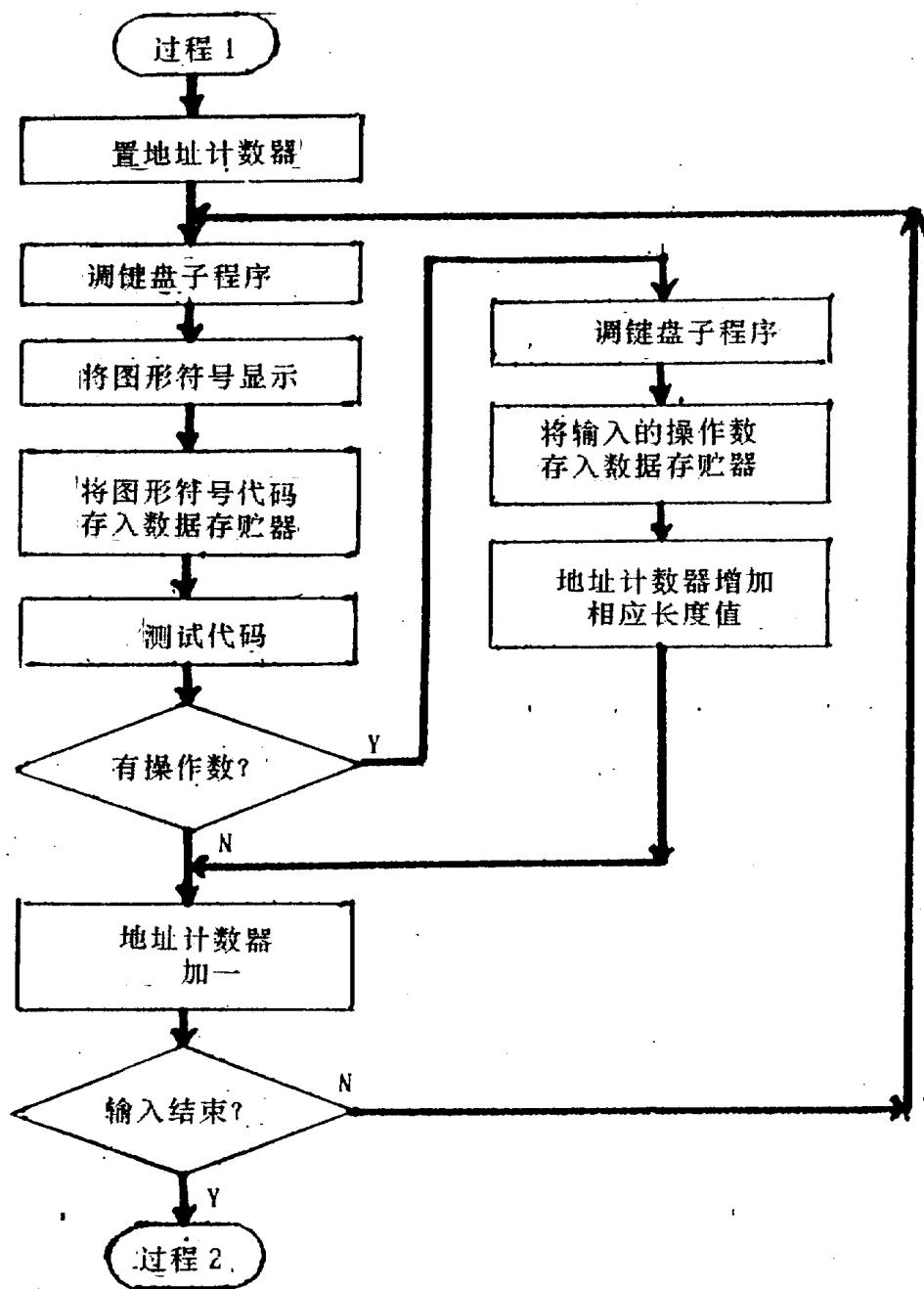


图 9

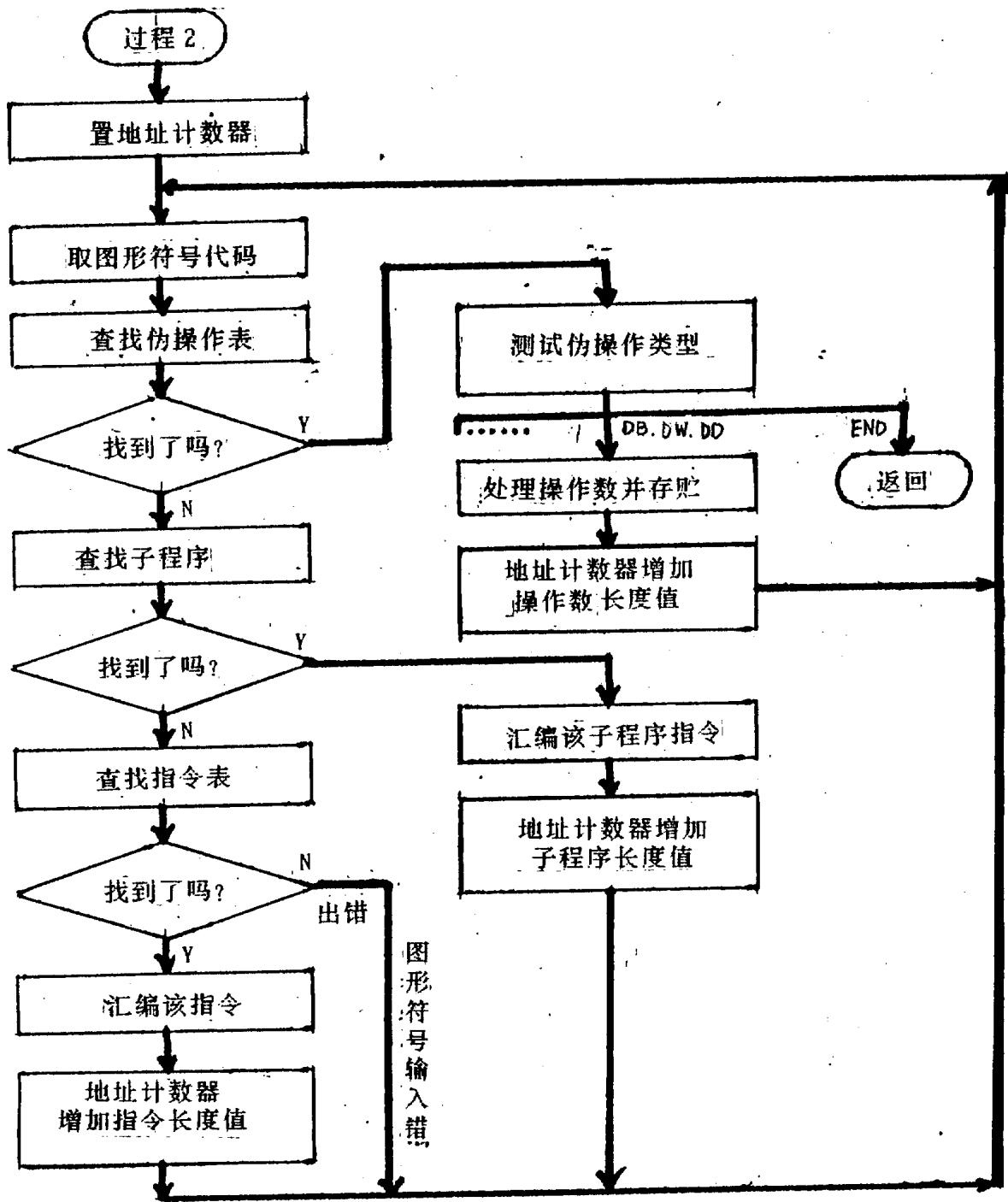


图 10