第10卷 第3期 华侨大学学报自然科学版 Vol.10 No.3
1989年7月 JOURNAL OF HUAQIAO UNIVERSITY (NATURAL SCIENCE) Jul. 1989

一种用于微机结构化汇编语言程序设计的语言

张 佳 峰

（计算中心）

摘 要

本文定义了一种用于微机结构化汇编语言程序设计的语言——伪高级语言，并举例说明其应用.

关键词 程序设计，汇编语言，微型计算机

一、引 言

用汇编语言编程，程序设计复杂，且难形成好的文件。要理解、熟悉一个非结构化汇编语言程序所花的代价有时甚至比重写一个还大，查错、修改和扩充就更困难了。结构化汇编语言程序设计的目的在于对程序的复杂性加以系统地控制，使程序结构层次分明，以提高编程效率。

二、结构化程序设计的基本原则

1．按 Dijkstra 的定义，结构化程序设计的基本原则是：(1) 程序块之间必须满足嵌套原则；(2) 每个程序块只能有一个入口和一个出口；(3) 程序的流程必须是单向的；(4) 程序块的结构只能是线性顺序结构、条件结构或循环结构〈或称前检查循环结构〉；(5) 和程序控制流相对应的，数据也必须是结构化的。

2．如果只用上述三种基本结构编程常会过于繁琐，故引入下述扩展：(1) 选择结构；(2) 后检查循环结构：在检查循环结束条件之前，先执行程序块；(3) 带 LEAVE 的循环结构〈或称中间检查循环结构〉：在执行一个循环期间若出现一种状态使得循环进一步执行变得无意义时，就直接离开循环。

三、用于微机结构化汇编语言程序设计的伪高级语言

一般常用结构图和树图作为结构化程序设计的表达方式。

本文1987年11月26日收到。

这里定义一种结构化汇编语言程序设计语言——伪高级语言作为表达程序结构的另一种方式。它满足上述 Dijkstra 定义的基本原则，并具有下述结构：

(1) 线性顺序结构：

```
( BLOCK NAME )
! block 1
! :
! :
! block n
( END BLOCK NAME )
```

(2) 条件结构：

```
IF C
! THEN
!   block 1
! ELSE
!   block 2
END IF
```

(3) 选择结构：

```
SELECT VE
! WHEN (W1)
!   block 1
! WHEN (W2)
!   block 2
! :
! :
! WHEN (Wn)
!   block n
! OTHER
!   block r
END SELECT
```

(4) 循环结构：

(i)前检查循环结构

```
DO
! WHILE C
!   block
END DO
```

(ii)带LEAVE的循环结构（或中间检查循环结构）：

```
DO
! block 1
! IF C
! ! THEN
! !   block 3
+ !   LEAVE(DO)
! END IF
! block 2
END DO
```

```
DO
! block 1
+IF C THEN LEAVE (DO)
! block 2
END DO
```

(iii)后检查循环结构：

```
DO
! block
! UNTIL C
END DO
```

此外，还可以是子程序结构：

```
SRNAME
! block 1
! :
! :
! block n
END SRNAME
```

为使程序层次分明，便于阅读、修改，必须严格按照上面给出的基本结构标准格式编写程序。在程序块名（BLOCK-NAME）、子程序名（SRNAME）或关键词（IF、SELECT、DO）和基本结构的结束（END）之间的各行之前依所处的层次写入数个（！），以便明了程序块、子程序或基本结构的范围。（！）越多，所处的层次越低。关键词 LEAVE 所在行的第一个（！）以（＋）代替，以表明在何处离开程序块。

下面以 EPROM 编程器写入子程序 SEPWR 为例，用结构图和树图（功能树）分别示于图 1、2。

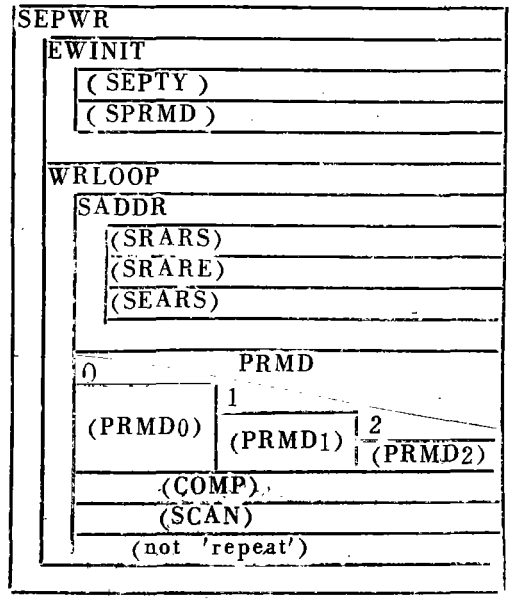

图 1

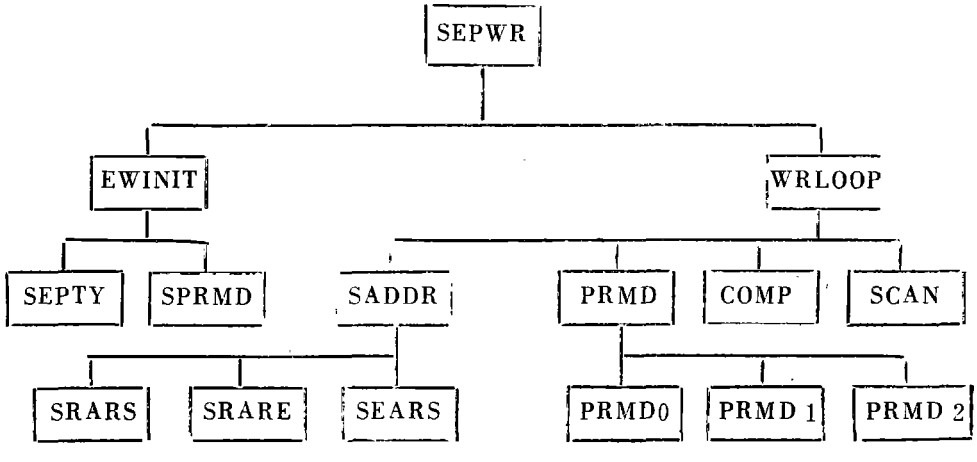

图2

用伪高级语言设计，写入子程序示下：

```
SEPWR
! EWINIT
! ! SEPTY
! ! SPRMD
! END EWINIT
! DO WRLOOP
! ! SADDR
! ! ! SRARS
! ! ! SRARE
! ! ! SEARS
! ! END SADDR
! ! SELECT PRMD
! ! ! WHEN( 0 )
! ! !   PRMD 0
! ! ! WHEN( 1 )
! ! !   PRMD 1
! ! ! WHEN( 2 )
! ! !   PRMD 2
! ! END SELECT PRMD
! ! COMP
! ! SCAN
! ! UNTIL( not `repeat ' )
! END DO WRLOOP
END SEPWR
```

```
SEPTY ( Subroutine )
! INITEP
! ! block
! END INITEP
! Do EPLOOP
! ! SCAN
! ! SELECT CHAR
! ! ! WHEN( `+ ' )
! ! !   IF counter < > selmax
! ! ! !   THEN
! ! ! !     forwd 1
! ! ! !   ELSE
! ! ! !     forwd 2
! ! !   END IF forwd
! ! ! WHEN( `- ' )
! ! !   IF counter < > 1
! ! ! !   THEN
! ! ! !     backw 1
! ! ! !   ELSE
! ! ! !     backw 2
! ! !   END IF backw
! ! ! WHEN( 'CR ' )
+! ! !   LEAVE( DO EPLOOP )
! ! ! OTHER( no operation )
! ! END SELECT CHAR
! END DO EPLOOP
END SEPTY
```

这里同时给出了其中的一个低级子程序 SEPTY。

由上可见，伪高级语言既能清楚地表示程序的动态结构，又能表示程序的静态结构；既能提供程序的梗概，也能部分地提供程序的细节，是一种比结构图和树图或许更好的表达方式。

四、伪高级语言的汇编语言编码

伪高级语言程序和相应的汇编语言程序在逻辑上和物理上应是完全一致的。由于使用了标准结构，可直接由前者写出后者，后者是前者的进一步精细化，而前者是后者的注释。

1．基本结构的汇编语言编码

在以下汇编语言中，OPS 代表能影响标志的语句，如算术、逻辑、比较或测试指令；OP 代表除转移指令以外的任何操作，也可是子程序调用；在 JUMP 指令中，NE 表示比较的两数不相等，而NOT C 表示不满足条件 C。

（1）线性顺序结构：由一系列线性排列的语句〈不包括转移语句〉组成。这种最简单结构可用程序块名和程序块结束把它自身与其他程序块分开，但一般也可略去。

（2）条件结构：

(i) IF-THEN结构:

```
        OPS                          ; IF C
        JUMP ,NOT C labeln           ; ! THEN
        OP                           ; !   block
        :                            ; !    :
        :                            ; !    :
        OP                           ; !    :
                                     ; END IF
                                     ; next block
labeln: …
```

(ii) IF-THEN-ELSE 结构:

```
        OPS                          ; IF C
        JUMP ,NOT C label 2          ; ! THEN
        OP                           ; !   block 1
        :                            ; !    :
        :                            ; !    :
        OP                           ; !    :
        JUMP     labeln              ; ! ELSE
label 2: OP                          ; !   block 2
        :                            ; !    :
        :                            ; !    :
        OP                           ; !    :
                                     ; END IF
                                     ; next block
labeln: …
```

(3) 选择结构:

```
        LOAD     A, VE               ; SELECT VE
        COMP     A, WI               ; !
        JUMP ,NE label 2             ; ! WHEN(W1)
        OP                           ; !   block 1
        :                            ; !    :
        :                            ; !    :
        OP                           ; !    :
        JUMP     labeln              ; !
label 2: COMP    A, W 2              ; !
        JUMP ,NE labelr              ; ! WHEN(W 2)
        OP                           ; !   block 2
        :                            ; !    :
        :                            ; !    :
        OP                           ; !    :
        JUMP     labeln              ; ! OTHER
labelr: OP                           ; !   block r
        :                            ; !    :
```

```
        :                                ; !  :
        OP                               ; !  :
                                         ; END SELECT
    labeln : …                           ; next block
```

只有在变量与所有可能值比较且不相等时才执行 OTHER 分支，通常它对应于出错情况。

（4）循环结构：

（i）前检查循环结构：

```
                DO-WHILE
    label : OPS                          ; DO
        JUMP ,NOT C labeln               ; ! WHILE C
        OP                               ; !   block
        :                                ; !  :
        :                                ; !  :
        OP                               ; !  :
        JUMP     label                   ; END DO
    labeln : …                           ; next block
```

（ii）带 LEAVE 的循环结构：

```
                                         ; DO
    label : OP                           ; ! block 1
        :                                ; !  :
        :                                ; !  :
        OP                               ; !  :
        OPS                              ; !
        JUMP ,C labeln                   ;   + IF C THEN LEAVE( DO )
        OP                               ; ! block 2
        :                                ; !  :
        :                                ; !  :
        OP                               ; !  :
        JUMP     label                   ; END DO
    labeln : …                           ; next block
```

或

```
                                         ; DO
  label 1 : OP                           ; ! block 1
        :                                ; !  :
        :                                ; !  :
        OP                               ; !  :
        OPS                              ; ! IF C
        JUMP ,NOT C label 2              ; ! ! THEN
        OP                               ; ! !   block 3
        :                                ; ! !  :
        :                                ; ! !  :
```

```
        OP                          ;  ! ! :
        JUMP labeln                 ;  + ! LEAVE(DO)
                                    ;  ! ENC IF
label 2 : OP                        ;  ! block 2
        :                           ;  ! :
        :                           ;  ! :
        OP                          ;  ! :
        JUMP label 1                ;  END DO
labeln : …                          ;  next block
```

(iii)后检查循环结构：

```
        DO-UNTIL
                                    ;  DO
label : OP                          ;  ! block
        :                           ;  ! :
        :                           ;  ! :
        OP                          ;  ! :
        OPS                         ;  ! UNTIL C
        JUMP ,NOT C label           ;  END DO
```

2. 程序块结束（END）的编码以及有关 LEAVE 和堆栈的问题

（1）各种结构程序块结束（END）的编码见表 1。

表 1

程序块类型	END 的编码
线性顺序结构	——
条件结构	——
选择结构	——
前检查循环结构	无条件转移至循环开始
带 LEAVE 的循环结构	无条件转移至循环开始
后检查循环结构	条件转移至循环开始
子程序	从子程序返回

如果子程序不使用堆栈或者推入和弹出操作的次数相同，那么子程序结束的编码即 RET。否则，就必须在执行 RET 之后使堆栈指针恢复到子程序开始时的值，以保证从子程序正确返回。

（2）在不同情况下，程序块中 LEAVE 的转移目的地也是不同的。

（i）在带 LEAVE 的循环结构中，LEAVE 意味着转移至该程序块结束（END）后的下一程序块的第一个语句，从而离开该程序块。

（ii）若利用 LEAVE 离开子程序，LEAVE 意味着转移至子程序的结束（END），在这里 END 包括堆栈指针的再生。

如果子程序不使用堆栈，则 LEAVE 可直接转移至 END 的编码 RET。

```
                                    ;  SPNAME(Subroutine name)
SRNAME : OP                         ;  ! block 1
```

```
        :                          ;  !  :
        :                          ;  !  :
        OP                         ;  !  :
        JUMP ,C label              ;  +  IF C THEN LEAVE( SRNAME )
        OP                         ;  !  block 2
        :                          ;  !  :
        :                          ;  !  :
        OP                         ;  !  :
label : RET                        ;     END SRNAME
```

如果子程序使用了堆栈，为免引起堆栈混乱，以从子程序正确返回，在子程序开始时先把堆栈指针保存在内存单元中，而 LEAVE 导致转移至 RET 前面的那个语句，该语句再生堆栈指针．注意，这里的子程序必须是非递归的．

```
                                   ;  SRNAME( Subroutine name )
SRNAME : LOAD MSP ,SP              ;  !  save SP
         OP                        ;  !  block 1
         :                         ;  !  :
         :                         ;  !  :
         OP                        ;  !  :
         JUMP ,C label             ;  +  IF C THEN LEAVE ( SRNAME )
         OP                        ;  !  block 2
         :                         ;  !  :
         :                         ;  !  :
         OP                        ;  !  :
 label : LOAD SP ,MSP              ;  !  restore SP
         RET                       ;     END SRNAME
```

如果利用 LEAVE 直接由低级程序块离开高级程序块，可以仿照上述方法处理．

3．伪高级语言在 Z80 汇编语言程序设计中的应用

下面以 EPROM 编程器写入子程序 SEPWR 中的一个低级子程序——选择 EPROM 类型子程序 SEPTY为例．在这里可供用户选择的 EPROM 共有八种．首先在 LED 显示器上显示第一种 EPROM 的名称，若正是用户使用的类型，按 'CR' 键，程序就把用户的选择存入指定内存单元，并返回主程序；否则，用户可用 '+' 或 '-' 键来显示第二种或第八种 EPROM，并依此找到所使用的类型．程序中调用了子程序 SCAN，它执行把显示缓冲器的内容送出显示并扫描键盘的循环，直至有键按下并识别之．

```
                                   ;  SEPTY
                                   ;  !  INITEP
SEPTY : LD HL, EPHEAD              ;  !  !  HL points to the first display pattern
        LD B, 1                    ;  !  !  counter=1
        LD DE, 6                   ;  !  !
                                   ;  !  END INITEP
```

```
                          ; ! DO EPLOOP
EPLOOP :CALLD SCAN        ; ! ! SCAN( display, scan the keyboard until
                          ; ! !   a button is pressed and find it out )
                          ; ! ! SELECT CHAR
        CP  '+'           ; ! !
        JR  NZ, EPL1      ; ! ! ! WHEN( '+' )
        CP  8             ; ! ! ! IF counter < > selmax ( selmax=8 )
        JR  Z, FORWD 2    ; ! ! ! ! THEN
        INC B             ; ! ! ! !   forwd 1 ( counter+1, HL points to
        ADD HL, DE        ; ! ! ! !     the next display pattern )
        JP  EPL 3         ; ! ! ! ! ELSE
FORWD 2:LD B, 1           ; ! ! ! !   forwd 2 ( counter=1, HL points to
        LD HL, EPHEAD     ; ! ! ! !     the first display pattern )
        JP  EPL 3         ; ! ! ! !
                          ; ! ! ! END IF forwd
EPL 1   :CP '-'           ; ! !
        JR  NZ, EPL 2     ; ! ! ! WHEN( '-' )
        CP  1             ; ! ! ! IF counter < > 1
        JR  Z, BACKW 2    ; ! ! ! ! THEN
        DEC B             ; ! ! ! !   backw 1 ( counter-1, HL points to
        AND A             ; ! ! ! !   the last display pattern )
        SBC HL, DE        ; ! ! ! !
        JP  EPL 3         ; ! ! ! ! ELSE
BACKW 2:LD B, 8           ; ! ! ! !   backw 2 ( counter=8, HL points to
        LD HL, EPHEAD+42  ; ! ! ! !   the least display pattern )
        JP  EPL 3         ; ! ! ! !
                          ; ! ! ! ! END IF backw
EPL 2   :CP "CR"          ; ! ! !
        JR NZ, EPL 3      ; ! ! ! WHEN( 'CR' )
        LD A, B           ; ! ! !
        LD ( INITFL ), A  ; ! ! ! store the result
        JP ENDSET         ; +! ! LEAVE( DO EPLOOP )
                          ; ! ! ! OTHER( no operation )
                          ; ! ! END SELECT CHAR
EPL 3:  JP SEPTY          ; ! END DO EPLOOP
ENDSET: RET               ;  END SEPTY
```

五、结 束 语

本文定义的伪高级语言具有以下特点。

(1) 由标准基本结构组成。用它编写的程序层次分明，可读性好，易于查错、修改和扩

充。

(2) 能较清楚地表示程序的动态和静态结构。

(3) 可直接由伪高级语言程序写出相应的汇编语言程序。程序清单稍加补充即形成较好的文件。

(4) 只额外使用了‘!’，易于编写和编辑，大大提高编程效率。

参　考　文　献

[1] Dijkstra, E.W. and Hoare, C.A., *Structured Programming*, Academic Press, (1972).

[2] Nicoud, J.D., Sommer, R. and Rothlisberger, H., Nur 70 Befehle, *Polyscope*, 4 (1977).

A Language for Use in Microcomputer Structured Assembly Language Programming

Zhang Jiafeng

Abstract

This paper defines a pseudo-level language for use in microcomputer structured assembly language programming, and exemplifies its usage.

Key words programming, assembly language, microcomputer