汇编程序示例

```
DATA SEGMENT    ; 数据段
     NUM    DW   0011101000000111B
     NOTES DB   'The result is :', '$'
DATA ENDS ; 数据段结束

STACK SEGMENT   ; 堆栈段
     STA   DB   50 DUP(?)
     TOP   EQU  LENGTH  STA
STACK ENDS; 堆栈段结束

CODE SEGMENT    ; 代码段
     ASSUME    CS:CODE, DS:DATA, SS:STACK
 BEGIN:
     MOV   AX, DATA
     MOV   DS, AX      ; 为DS赋初值
```

汇编程序示例（续1）

```
        MOV   AX, STACK
        MOV   SS, AX                    ; 为SS赋初值
        MOV   AX, TOP
        MOV   SP, AX                    ; 为SP赋初值
        MOV   DX, OFFSET NOTES          ; 显示提示信息
        MOV   AH, 9H
        INT   21H
        MOV   BX, NUM                   ; 将数装入BX
        MOV   CH, 4                     ; CH作循环计数器
ROTATE :
        MOV   CL, 4                     ; CL中放移位位数
        ROL   BX, CL
        MOV   AL, BL
        AND   AL, 0FH                   ; AL中为一个16进制数
```

汇编程序示例（续2）

```
        ADD   AL, 30H              ; 转换为ASCII码值
        CMP   AL, '9'              ; 是0~9的数码?
        JLE   DISPLAY
        ADD   AL, 07H              ; 在A~F之间
DISPLAY:
        MOV   DL, AL               ; 显示16进制数
        MOV   AH, 2
        INT   21H
        DEC   CH
        JNZ   ROTATE
        MOV   AX, 4C00H            ; 返回
        INT   21H
CODE    ENDS                       ; 代码段结束
        END   BEGIN                ; 模块结束
```

汇编语言程序的组成

- 分段结构
  - 按段进行组织，最多由4个段组成（代码、数据、附加、堆栈）
  - 每个段以“段名 SEGMENT”开始，以“段名 ENDS”结束
- 语句行
  - 段由若干语句行组成
  - 语句行的三种类型：指令、伪指令、宏指令

```
DATA SEGMENT
      … ; 数据段语句
DATA ENDS

STACK SEGMENT
      … ; 堆栈段语句
STACK ENDS

CODE SEGMENT
      … ; 代码段语句
CODE ENDS
```

指令语句（可执行语句）

- 格式

  [标号:] 指令操作助记符 [操作数表达式1 [, 操作数表达式2]] [; 注释]

- 说明
  - 指令操作助记符（指令名）是不可缺少的主体
  - 方括号中的内容根据需要可省略
  - 注释以分号开头，将被汇编器忽略

- 示例

  L: ADD AX, BX ; 这是一条指令语句

伪指令语句（说明性语句）

- 格式

  [名字] 伪指令指示符 [操作数表达式1 [, 操作数表达式2 [, ……]]] [; 注释]

- 说明
  - 伪指令指示符（伪指令名）是不可缺少的主体
  - 方括号中的内容根据需要可省略
  - 名字可为段名、过程名、变量名、符号名（或常量名）、宏名、结构名、记录名等

- 示例

  A DB 20H, 30H ; 声明字节变量的伪指令语句

语句的执行

- 伪指令语句的执行
  - 汇编器计算伪指令语句中表达式的值
  - 不产生机器代码
  - 汇编器解释伪指令语句的含义并遵照“执行”
- 指令语句的执行
  - 汇编器计算指令语句中表达式的值
  - 汇编器将指令语句翻译成机器指令代码
  - 程序运行时，由CPU按机器指令代码的要求完成各种运算与操作

伪指令语句1. 数据定义

- 变量
  - 编程时只能确定其初始值，程序运行期间可修改其值的数据对象称为变量
  - 变量是存储单元中的数据，可定义在任何段，但通常都定义在数据段（DS）和附加段（ES）
  - 变量由伪指令说明符DB、DW、DD等定义
- 示例

```
A   DB   50, 60, 70, 80
    DW   50, 60, 70, 80
    DD   50, 60, 70, 80
```

“变量名”就是变量地址的名字，也可称为“变量的符号地址”

伪指令语句2. 符号定义

- 常数、常量
  - 编程时已经确定其值，程序运行期间不会改变其值的数据对象称为常数
  - 常数表达式的名字称为常量
  - 常量可用伪指令说明符“EQU”或“=”定义
  - 常量不产生目标代码，不占用存储单元
- 示例

```
A  EQU  7 ;
A  EQU  8 ; 错误，  “EQU”左边的符号名不可重复定义
B = 7       ;
B = 8       ; 正确，  “=”左边的符号名可以重复定义
```

数据定义和符号定义的示例

```
DATA  SEGMENT     ; 数据段
      W1    DW    1, 2, 3, 4, 5, 6, 7
      B1    DB    10, 20, 30, 40, 50
      N1    EQU   B1-W1        ; N1=14
DATA  ENDS        ; 数据段结束
CODE  SEGMENT     ; 代码段
      ASSUME   CS:CODE, DS:DATA
   BEGIN:
      …
      MOV    AX,   W1
      MOV    B1,   AL
      …
CODE    ENDS      ; 代码段结束
      END    BEGIN      ; 模块结束
```

伪指令语句3. 段定义

- 段定义说明符1：SEGMENT（段开始）
  - 示例

  CODE　SEGMENT
- 段定义说明符2：ENDS（段结束）
  - 示例

  CODE　ENDS
- 段定义说明符3：ASSUME（指定段寄存器）
  - 示例

  ASSUME　CS:CODE，DS:DATA，SS:STACK

示例

```
DATA    SEGMENT                          ; 数据段
        NUM      DW   0011101000000111B
        NOTES    DB   'The result is :', '$'
DATA    ENDS                             ; 数据段结束
CODE    SEGMENT                          ; 代码段
        ASSUME   CS:CODE, DS:DATA
    BEGIN:
        MOV   AX, DATA
        MOV   DS, AX                     ; 为DS赋初值
        …

CODE    ENDS                             ; 代码段结束
        END   BEGIN                      ; 模块结束
```

伪指令语句4. 指定段内的偏移地址

- ORG说明符
  - 格式：ORG　常数表达式
  - 作用：指定当前可用的存储单元的偏移地址为常数表达式的值

- EVEN说明符
  - 格式：EVEN
  - 作用：将当前可用的存储单元的偏移地址调整为最近的偶数值

示例

```
DATA    SEGMENT
        ORG   1000H
        A DB   47H, 12H, 45H
        EVEN
        B DB   47H
DATA    ENDS
```

说明：

① ORG指令将A的偏移地址部分指定为1000H

② 从A开始存放3个字节变量，占用地址1000H、1001H和1002H

③ EVEN指令会将B的偏移地址部分从1003H调整为偶数地址1004H

伪指令语句5. 过程定义

- PROC说明符
  - 格式：过程名　PROC　类型属性名
  - 说明：从“过程名”代表的地址开始定义一个过程；“类型属性名”可选择NEAR（近过程）或FAR（远过程），默认为NEAR

- ENDP说明符
  - 格式：过程名　ENDP
  - 说明：表示该过程到此结束。此处的“过程名”必须与过程开始时PROC左边的“过程名”相同

示例

```
…
CODE    SEGMENT                    ; 代码段
        ASSUME  CS:CODE, DS:DATA
    BEGIN:
        …
        CALL    DISPLAY            ; 过程调用
        …

        DISPLAY   PROC   NEAR ; 过程定义
                …
                RET
        DISPLAY   ENDP
CODE    ENDS                       ; 代码段结束
        END   BEGIN                ; 模块结束
```

宏定义

- 宏（MACRO）
  - 宏是源程序中一段有独立功能的程序代码
  - 宏只需在源程序中定义一次
  - 源程序中可以多次使用宏指令来调用宏
- 宏定义
  - 用一组伪指令MACRO和ENDM来实现

```
宏指令名   MACRO  [ 形式参数, 形式参数, …]
                …
                … ；此间的指令序列称为“宏定义体”
                ENDM
```

宏调用和宏展开

宏调用

- 在源程序中调用宏指令，称宏调用

宏展开

① 用宏定义体取代源程序中的宏指令名

② 用实在参数取代宏定义的形式参数

- 当源程序被汇编时，汇编器将对每个宏调用作宏展开

示例

```
…
MOV    AX, 1234H
SaveReg
ADD    AX, BX
SaveReg
…
```

宏调用

```
SaveReg    MACRO
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
ENDM
```

宏定义

```
…
MOV    AX, 1234H
+ PUSH    AX
+ PUSH    BX
+ PUSH    CX
+ PUSH    DX
ADD    AX, BX
+ PUSH    AX
+ PUSH    BX
+ PUSH    CX
+ PUSH    DX
…
```

宏展开

示例2

宏调用

```
…
MOV    AL, 12H
SHIFT 4, AL, AL
MOV    BX, 5678H
SHIFT 6, BX, AR
…
```

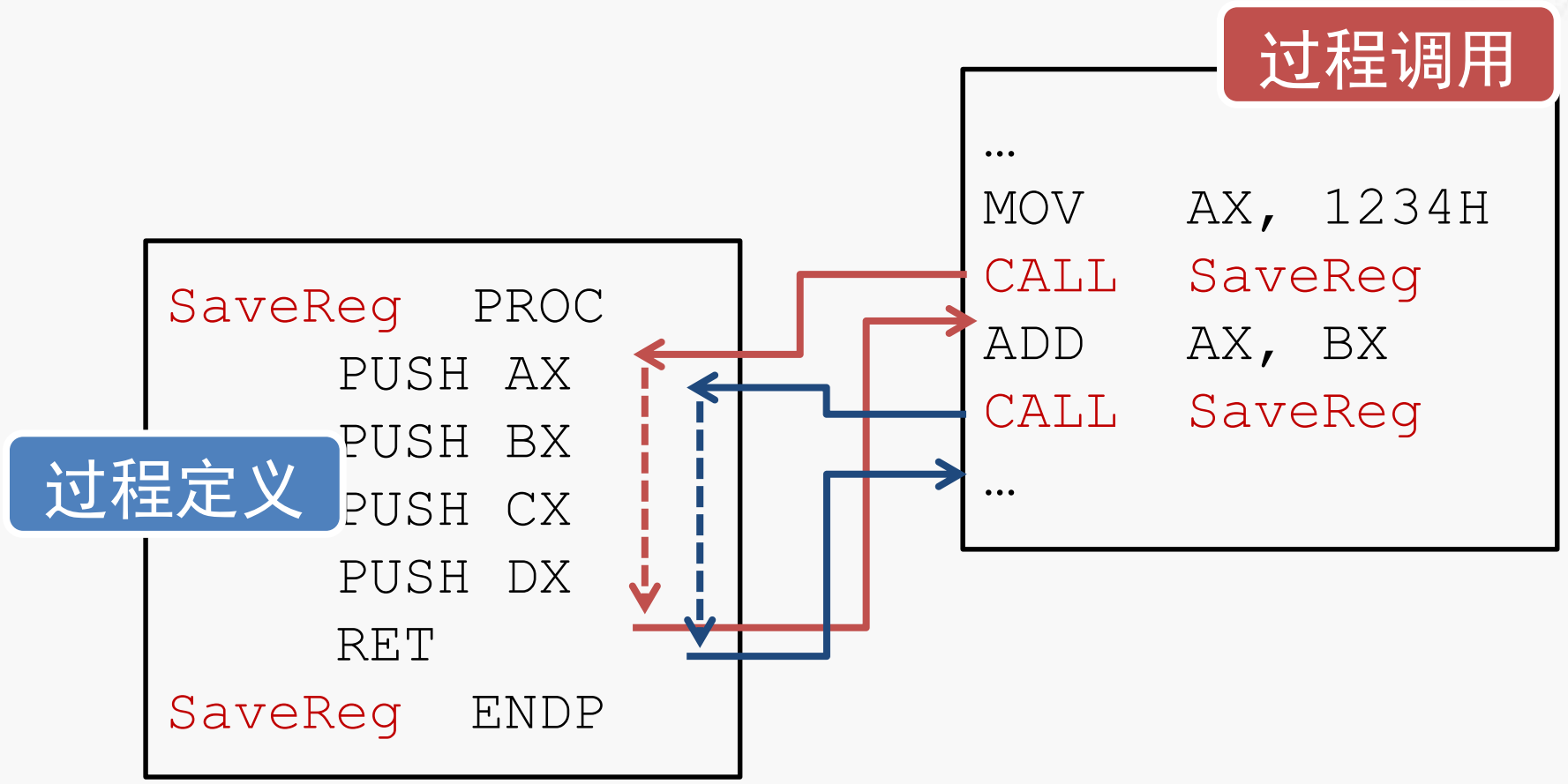
宏展开

```
…
MOV   AL, 12H
+ MOV    CL, 4
+ SAL    AL, CL
MOV   BX, 5678H
+ MOV    CL, 6
+ SAR    BX, CL
…
```

```
SHIFT    MACRO   X, Y, Z
      MOV    CL, X
      S&Z     Y, CL
ENDM
```

宏定义

过程调用的执行流程


过程调用
…
MOV AX, 1234H
CALL SaveReg
ADD AX, BX
CALL SaveReg
…
过程定义
SaveReg PROC
PUSH AX
PUSH BX
PUSH CX
PUSH DX
RET
SaveReg ENDP

过程调用和宏调用的区别

- 区别1：在处理时间上不同
  - 宏指令：在汇编时展开，执行速度快
  - 过程：在执行时调用，执行速度慢

- 区别2：传递参数的方式不同
  - 宏指令：用实在参数替代形式参数
  - 过程：使用通用寄存器、堆栈等

过程调用和宏调用的区别（续）

- 区别3：代码长度不同
  - 宏指令
    - 目标代码长，占内存空间大
    - 宏调用的次数越多，所占内存空间越大
  - 过程
    - 目标代码短，占内存空间小
    - 所占内存空间不会随调用次数的增加而增加