# 8086 汇编常用函数库手册

这份文档整理了 `常用函数list` 目录下的所有汇编函数，方便考试和实验时快速查阅和使用。

## 1. 快速使用指南

### 引入方法

所有函数已经打包在 `LIB.ASM` 中。你只需要在你的代码 `CODE SEGMENT` 的末尾（`CODE ENDS` 之前）添加一行 `INCLUDE` 指令即可使用所有函数。

```
INCLUDE 常用函数list\LIB.ASM
```

### 标准考试模板（`CLEAN_SKELETON.asm`）

可以直接复制以下模板开始编写程序：

```
STACK SEGMENT
            DB 256 DUP(0)
STACK ENDS

DATA SEGMENT
      ; 在这里定义数据
      ; MSG DB 'HELLO, WORLD!$'
DATA ENDS

CODE SEGMENT
            ASSUME  CS:CODE, DS:DATA, SS:STACK
      START:
            MOV     AX, DATA
            MOV     DS, AX

      ; ==========================================
      ; 在这里编写你的主程序代码
      ; ==========================================

      ; 退出程序
            MOV     AX, 4C00H
            INT     21H

      ; ==========================================
      ; 引入所有常用函数库
      ; ==========================================
            INCLUDE 常用函数list\LIB.ASM

CODE ENDS
END START
```

---

# 2. 输入输出函数 (I/O)

## PUTC

- **功能**: 打印一个字符。
- **输入**: 栈参数 `[BP+4]` (字符 ASCII 码)。
- **注意**: 调用前需 `PUSH` 字符，调用后需 `ADD SP, 2` 平栈。

```
PUTC PROC
             PUSH    BP                      ; 保存调用者的基指针
             MOV     BP,SP                   ; 设置当前栈帧的基指针
             PUSH    DX                      ; 保存 DX 寄存器的值
             PUSH    AX                      ; 保存 AX 寄存器的值

             MOV     DL,[BP-2]               ; 从栈中获取函数参数
             MOV     AH,02H                  ; DOS 02H 功能调用
             INT     21H

             POP     AX
             POP     DX
             POP     BP
             RET
PUTC ENDP
```

## PUTS

- **功能**: 打印以 `$` 结尾的字符串。
- **输入**: 栈参数 `[BP+4]` (字符串偏移地址 DX)。
- **注意**: 调用前需 `PUSH` 字符串偏移地址 (OFFSET MSG)，调用后需 `ADD SP, 2` 平栈。

```
PUTS PROC
             PUSH    BP
             MOV     BP,SP
             PUSH    DX
             PUSH    AX

             MOV     DX,[BP-2]
             MOV     AH,09H                  ; DOS 09H 功能调用
             INT     21H

             POP     AX
             POP     DX
             POP     BP
             RET
PUTS ENDP
```

## GETS

- **功能**: 从键盘读入一个字符串到指定缓冲区。
- **输入**: 栈参数 `[BP+4]` (缓冲区首地址 DX)。
- **注意**: 缓冲区格式需符合 DOS 0AH 功能调用要求（第一个字节为缓冲区大小）。

```
GETS PROC
             PUSH    BP
             MOV     BP,SP
             PUSH    DX
             PUSH    AX

             MOV     DX,[BP-2]
             MOV     AH,0AH                  ; DOS 0AH 功能调用
             INT     21H

             POP     AX
             POP     DX
             POP     BP
             RET
GETS ENDP
```

## NLINE

- **功能**: 输出换行符 (CR + LF)。
- **输入**: 无。

```
NLINE PROC
             PUSH    BP
             MOV     BP,SP
             PUSH    DX

             MOV     DX,0DH
             PUSH    DX
             CALL    PUTC
             ADD     SP,2

             MOV     DX,0AH
             PUSH    DX
             CALL    PUTC
             ADD     SP,2

             POP     DX
             POP     BP
             RET
NLINE ENDP
```

---

# 3. 进制转换函数

## BIN_TO_DEC

- **功能**: 将 AX 中的二进制数转为 10 进制 ASCII 码并直接打印。
- **输入**: `AX` (数值)。
- **输出**: 无 (直接打印到屏幕)。

```
BIN_TO_DEC PROC NEAR
        BEG:
                    MOV     BX,10
                    MOV     CX,0
        LAST:       MOV     DX,0
                    DIV     BX
                    PUSH    DX
                    INC     CX
                    CMP     AX,0
                    JNZ     LAST

        AGA:        POP     DX
                    ADD     DX,30H
                    PUSH    DX
                    CALL    PUTC
                    ADD     SP,2
                    LOOP    AGA
                    RET
BIN_TO_DEC ENDP
```

## DEC_TO_BIN

- **功能**: 从键盘读取十进制数并转换为二进制存入 BX。
- **输入**: 无 (键盘输入)。
- **输出**: `BX` (数值)。

```
DEC_TO_BIN PROC NEAR
                    MOV     BX, 0
        NEWCHAR:
                    MOV     AH, 1
                    INT     21H

                    SUB     AL, 30H
                    JL      EXIT
                    CMP     AL, 9D
                    JG      EXIT

                    CBW
                    XCHG    AX, BX
                    MOV     CX, 10D
                    MUL     CX
                    XCHG    AX, BX
                    ADD     BX, AX
```

```
                JMP     NEWCHAR

        EXIT:
                RET
DEC_TO_BIN ENDP
```

## DEC_TO_HEX

- **功能**: 从键盘读取十进制数，并直接打印其十六进制表示。
- **输入**: 无 (从键盘读取)。
- **输出**: 打印十六进制数。

```
DEC_TO_HEX PROC NEAR
                PUSH    AX
                PUSH    BX
                PUSH    CX
                PUSH    DX

    ; 1. 读取十进制数到 BX
                MOV     BX, 0
    READ_DEC_LOOP:
                MOV     AH, 1
                INT     21H

                CMP     AL, 0DH         ; 回车结束
                JE      READ_DONE
                SUB     AL, 30H
                JL      READ_DONE       ; 非数字结束
                CMP     AL, 9
                JG      READ_DONE       ; 非数字结束

                CBW
                XCHG    AX, BX
                MOV     CX, 10
                MUL     CX
                ADD     BX, AX
                JMP     READ_DEC_LOOP

    READ_DONE:
                MOV     AH, 2
                MOV     DL, 0DH
                INT     21H
                MOV     DL, 0AH
                INT     21H

    ; 2. 打印 BX 为十六进制
                MOV     CH, 4           ; 4个十六进制位
    PRINT_HEX_LOOP:
                MOV     CL, 4
                ROL     BX, CL          ; 循环左移4位
                MOV     AL, BL
                AND     AL, 0FH         ; 取低4位
                ADD     AL, 30H
                CMP     AL, 39H
```

```
                JLE     IS_DIGIT_HEX
                ADD     AL, 7               ; 'A'-'F'
    IS_DIGIT_HEX:
                MOV     DL, AL
                MOV     AH, 2
                INT     21H
                DEC     CH
                JNZ     PRINT_HEX_LOOP

                POP     DX
                POP     CX
                POP     BX
                POP     AX
                RET
DEC_TO_HEX ENDP
```

## PUTDB16

- **功能**: 将 AX 中的 16 位二进制数以 16 进制形式打印。
- **输入**: `AX` (16位数值)。
- **输出**: 无 (打印4位16进制字符)。

```
PUTDB16 PROC
    PUSH    BX
    PUSH    CX
    PUSH    DX
    PUSH    AX

    MOV     BX, AX              ; 将AX的值移入BX进行处理

    MOV     CH, 04H             ; 设置循环次数为4次
SWITCHLAST:
    MOV     CL, 4H              ; 设置每次循环左移的位数为4位
    ROL     BX, CL              ; 将BX寄存器的内容左移4位
    MOV     AL, BL              ; 将左移后的BX寄存器的低8位加载到AL寄存器
    AND     AL, 0FH             ; 只保留AL寄存器的低4位
    ADD     AL, 30H             ; 转换为字符
    CMP     AL, 3AH
    JL      LAST
    ADD     AL, 07H             ; 'A'-'F'
LAST:
    MOV     DX, 00H
    MOV     DL, AL
    PUSH    DX
    CALL    PUTC
    ADD     SP, 2

    DEC     CH
    JNZ     SWITCHLAST

    POP     AX
    POP     DX
    POP     CX
    POP     BX
```

```
    RET
PUTDB16 ENDP
```

## PUTDB8

- **功能**: 将 AL 中的 8 位二进制数以 16 进制形式打印。
- **输入**: `AL` (8位数值)。
- **输出**: 无 (打印2位16进制字符)。

```
PUTDB8 PROC
            PUSH    BX
            PUSH    CX
            PUSH    DX
            PUSH    AX

            MOV     BL, AL                      ; 将AL移入BL
            MOV     BH, AL
            MOV     CH, 02H                     ; 设置循环计数器
    SWITCH:
            MOV     CL, 4H
            ROL     BL, CL
            MOV     AL, BL
            AND     AL, 0FH
            ADD     AL, 30H
            CMP     AL, 39H
            JLE     PRINT
            ADD     AL, 07H

    PRINT:
            MOV     DX, 00H
            MOV     DL, AL
            PUSH    DX
            CALL    PUTC
            ADD     SP, 2

            DEC     CH
            JNZ     SWITCH

            POP     AX
            POP     DX
            POP     CX
            POP     BX
            RET
PUTDB8 ENDP
```

## HEX_STR_TO_WORD

- **功能**: 将 4 位 16 进制字符串转为数值。
- **输入**: `DS:SI` 指向字符串 (例如 "1A2B")。
- **输出**: `AX` = 数值 (例如 1A2BH), `CF=1` 表示出错。

```
HEX_STR_TO_WORD PROC NEAR
```

```
                PUSH    BX
                PUSH    CX
                PUSH    DX
                PUSH    SI

                XOR     BX, BX
                MOV     CX, 4

    HEX_LOOP:
                MOV     AL, [SI]
                CMP     AL, '0'
                JB      HEX_ERROR
                CMP     AL, '9'
                JBE     IS_DIGIT
                CMP     AL, 'A'
                JB      HEX_ERROR
                CMP     AL, 'F'
                JBE     IS_UPPER
                CMP     AL, 'A'
                JB      HEX_ERROR
                CMP     AL, 'F'
                JBE     IS_LOWER
                JMP     HEX_ERROR

    IS_DIGIT:
                SUB     AL, '0'
                JMP     ADD_VAL
    IS_UPPER:
                SUB     AL, 'A' - 10
                JMP     ADD_VAL
    IS_LOWER:
                SUB     AL, 'A' - 10
                JMP     ADD_VAL

    ADD_VAL:
                SHL     BX, 1
                SHL     BX, 1
                SHL     BX, 1
                SHL     BX, 1
                XOR     AH, AH
                ADD     BX, AX

                INC     SI
                LOOP    HEX_LOOP

                MOV     AX, BX
                CLC
                JMP     HEX_EXIT

    HEX_ERROR:
                STC

    HEX_EXIT:
                POP     SI
                POP     DX
                POP     CX
```

```
                    POP     BX
                    RET
HEX_STR_TO_WORD ENDP
```

---

# 4. 数学运算函数 (MATH_OPS)

## ADD_16 / SUB_16 / MUL_16 / DIV_16

- **ADD_16**: `AX = AX + BX`
- **SUB_16**: `AX = AX - BX`
- **MUL_16**: `DX:AX = AX * BX` (无符号)
- **DIV_16**: `AX = AX / BX` (商), `DX = AX % BX` (余数) (无符号)
- **SIGNED_DIV_16**: `AX = AX / BX` (商), `DX = AX % BX` (余数) (有符号)

```
ADD_16 PROC NEAR
                    ADD     AX, BX
                    RET
ADD_16 ENDP

SUB_16 PROC NEAR
                    SUB     AX, BX
                    RET
SUB_16 ENDP

MUL_16 PROC NEAR
                    MUL     BX
                    RET
MUL_16 ENDP

DIV_16 PROC NEAR
                    DIV     BX
                    RET
DIV_16 ENDP

SIGNED_DIV_16 PROC NEAR
                    CWD
                    IDIV    BX
                    RET
SIGNED_DIV_16 ENDP
```

---

# 5. 字符串处理函数 (STR_TOOLS)

## STRLEN

- **功能**: 计算以 `$` 结尾的字符串长度。
- **输入**: `DS:SI` 指向字符串。
- **输出**: `CX` = 长度。

```

```

```
STRLEN PROC NEAR
                    PUSH    SI
                    PUSH    AX
                    XOR     CX, CX
    STRLEN_LOOP:
                    MOV     AL, [SI]
                    CMP     AL, '$'
                    JE      STRLEN_END
                    INC     CX
                    INC     SI
                    JMP     STRLEN_LOOP
    STRLEN_END:
                    POP     AX
                    POP     SI
                    RET
STRLEN ENDP
```

## STRCMP

- **功能**: 比较两个字符串 (以 `$` 结尾)。
- **输入**: `DS:SI` = 字符串1, `ES:DI` = 字符串2。
- **输出**: `ZF=1` (相等), `ZF=0` (不等)。

```
STRCMP PROC NEAR
                    PUSH    AX
                    PUSH    SI
                    PUSH    DI
    STRCMP_LOOP:
                    MOV     AL, [SI]
                    CMP     AL, ES:[DI]
                    JNE     STRCMP_DIFF
                    CMP     AL, '$'
                    JE      STRCMP_EQUAL
                    INC     SI
                    INC     DI
                    JMP     STRCMP_LOOP

    STRCMP_DIFF:
                    JMP     STRCMP_EXIT

    STRCMP_EQUAL:
                    CMP     BYTE PTR ES:[DI], '$'

    STRCMP_EXIT:
                    POP     DI
                    POP     SI
                    POP     AX
                    RET
STRCMP ENDP
```

## STRSTR

- **功能**: 在主串中查找子串 (暴力匹配)。
- **输入**: `DS:SI` = 主串 (以 `$` 结尾), `DS:BX` = 子串 (以 `$` 结尾)。
- **输出**: `AX` = 子串在主串中的偏移地址 (找到), `FFFFH` (未找到)。

```
STRSTR PROC NEAR
                    PUSH    CX
                    PUSH    SI
                    PUSH    BX
                    PUSH    DX

                    MOV     DX, SI

    STRSTR_OUTER:
                    MOV     AL, [SI]
                    CMP     AL, '$'
                    JE      STRSTR_NOT_FOUND

                    PUSH    SI
                    PUSH    BX

    STRSTR_INNER:
                    MOV     AL, [SI]
                    MOV     AH, [BX]
                    CMP     AH, '$'
                    JE      STRSTR_FOUND
                    CMP     AL, AH
                    JNE     STRSTR_NEXT
                    CMP     AL, '$'
                    JE      STRSTR_NEXT

                    INC     SI
                    INC     BX
                    JMP     STRSTR_INNER

    STRSTR_NEXT:
                    POP     BX
                    POP     SI
                    INC     SI
                    JMP     STRSTR_OUTER

    STRSTR_FOUND:
                    POP     BX
                    POP     AX                      ; 这里的AX其实是栈里的SI (匹配起
始位置)
                    POP     DX                      ; 恢复DX
                    POP     BX
                    POP     SI
                    POP     CX
                    RET

    STRSTR_NOT_FOUND:
                    MOV     AX, 0FFFFH
```

```
                    POP     DX
                    POP     BX
                    POP     SI
                    POP     CX
                    RET
STRSTR ENDP
```

---

# 6. 排序函数 (SORT)

## BUBBLE_SORT

- **功能**: 对字节数组进行冒泡排序 (升序)。
- **输入**: `DS:SI` = 数组首地址, `CX` = 数组长度。
- **输出**: 数组被原地排序。

```
BUBBLE_SORT PROC NEAR
                PUSH    AX
                PUSH    BX
                PUSH    CX
                PUSH    DX
                PUSH    SI

                CMP     CX, 1
                JLE     SORT_DONE

                DEC     CX

    OUTER_LOOP:
                PUSH    CX
                PUSH    SI

    INNER_LOOP:
                MOV     AL, [SI]
                MOV     AH, [SI+1]
                CMP     AL, AH
                JLE     NO_SWAP

                MOV     [SI], AH
                MOV     [SI+1], AL

    NO_SWAP:
                INC     SI
                LOOP    INNER_LOOP

                POP     SI
                POP     CX
                LOOP    OUTER_LOOP

    SORT_DONE:
                POP     SI
                POP     DX
                POP     CX
                POP     BX
```

```
                POP     AX
                RET
BUBBLE_SORT ENDP
```