汇编语言中的数字输入输出（DOS INT 21H）

DOS的 `INT 21H` 中断本身没有直接的**数字输入输出**功能。数字的输入输出需要我们自己编写代码，在**ASCII字符串**和**二进制数值**之间进行转换。

一、数字处理的基本原理

在汇编中，数字以两种形式存在：

1. **二进制形式**：CPU可以直接运算，如 `AX=1234`
2. **ASCII形式**：用于显示，如 `"1234"`（31H, 32H, 33H, 34H）

转换流程：

```
输入："1234"（ASCII）→ 转换 → 1234（二进制）→ 运算
输出：1234（二进制）→ 转换 → "1234"（ASCII）→ 显示
```

二、字符串到数字的转换（输入）

1. 十进制字符串转数字（无符号）

```
; 输入：SI = 字符串地址（十进制ASCII）
; 输出：AX = 数字
; 算法：result = result × 10 + (digit - '0')

STR_TO_NUM PROC
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH SI

    XOR AX, AX          ; 结果清零
    MOV BX, 10          ; 基数10

CONVERT_LOOP:
    MOV CL, [SI]        ; 取一个字符
    CMP CL, '0'         ; 检查是否数字
    JB CONVERT_DONE
    CMP CL, '9'
    JA CONVERT_DONE

    SUB CL, '0'         ; ASCII转数字
    MUL BX              ; AX = AX × 10
    ADD AL, CL          ; 加上新数字
    ADC AH, 0           ; 处理进位

    INC SI
    JMP CONVERT_LOOP

CONVERT_DONE:
    POP SI
    POP DX
    POP CX
```

```
    POP BX
    RET
STR_TO_NUM ENDP
```

2. 十进制字符串转数字（带符号）

```
; 输入：SI = 字符串地址，可能以'-'开头
; 输出：AX = 数字（有符号），CF=1表示错误

STR_TO_NUM_SIGNED PROC
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH SI

    XOR AX, AX          ; 结果清零
    XOR CX, CX          ; CX=0表示正数，CX=1表示负数
    MOV BX, 10

    ; 检查符号
    MOV CL, [SI]
    CMP CL, '-'
    JNE POSITIVE
    MOV CX, 1           ; 设置负数标志
    INC SI              ; 跳过负号

POSITIVE:
CONVERT_LOOP2:
    MOV DL, [SI]
    CMP DL, '0'
    JB CONVERT_DONE2
    CMP DL, '9'
    JA CONVERT_DONE2

    SUB DL, '0'
    PUSH DX             ; 保存数字
    MUL BX              ; AX × 10
    POP DX
    ADD AL, DL
    ADC AH, 0

    INC SI
    JMP CONVERT_LOOP2

CONVERT_DONE2:
    ; 处理符号
    TEST CX, CX
    JZ POSITIVE_NUM
    NEG AX              ; 负数：取补码

POSITIVE_NUM:
    CLC                 ; 清除错误标志
    POP SI
    POP DX
    POP CX
```

```
    POP BX
    RET

ERROR:
    STC                 ; 设置错误标志
    POP SI
    POP DX
    POP CX
    POP BX
    RET
STR_TO_NUM_SIGNED ENDP
```

3. 十六进制字符串转数字

```
; 输入：SI = 字符串地址（十六进制，如"1A3F"）
; 输出：AX = 数字

HEX_STR_TO_NUM PROC
    PUSH BX
    PUSH CX
    PUSH SI

    XOR AX, AX          ; 结果清零
    MOV BX, 16          ; 基数16

HEX_LOOP:
    MOV CL, [SI]
    CMP CL, '0'
    JB HEX_DONE
    CMP CL, '9'
    JBE IS_DIGIT
    CMP CL, 'A'
    JB HEX_DONE
    CMP CL, 'F'
    JBE IS_UPPER
    CMP CL, 'a'
    JB HEX_DONE
    CMP CL, 'f'
    JA HEX_DONE

IS_LOWER:
    SUB CL, 'a' - 10    ; 'a'=97, 转换为10
    JMP ADD_DIGIT

IS_UPPER:
    SUB CL, 'A' - 10    ; 'A'=65, 转换为10
    JMP ADD_DIGIT

IS_DIGIT:
    SUB CL, '0'         ; '0'=48

ADD_DIGIT:
    MUL BX              ; AX × 16
    ADD AL, CL
    ADC AH, 0
```

```
    INC SI
    JMP HEX_LOOP

HEX_DONE:
    POP SI
    POP CX
    POP BX
    RET
HEX_STR_TO_NUM ENDP
```

三、数字到字符串的转换（输出）

1. 数字转十进制字符串（无符号）

```
; 输入：AX = 数字
; 输出：DI = 缓冲区地址（以'$'结尾）

NUM_TO_STR PROC
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH DI

    MOV CX, 0           ; 数字位数计数器
    MOV BX, 10          ; 除数10

    ; 处理0的情况
    TEST AX, AX
    JNZ DIVIDE
    MOV BYTE PTR [DI], '0'
    MOV BYTE PTR [DI+1], '$'
    JMP CONV_DONE

DIVIDE:
    XOR DX, DX
    DIV BX              ; DX:AX ÷ 10
    PUSH DX             ; 保存余数（0-9）
    INC CX              ; 位数+1
    TEST AX, AX         ; 商是否为0？
    JNZ DIVIDE          ; 不为0则继续

    ; 弹出并转换为ASCII
POP_LOOP:
    POP AX
    ADD AL, '0'         ; 转换为ASCII
    MOV [DI], AL
    INC DI
    LOOP POP_LOOP

    MOV BYTE PTR [DI], '$'  ; 添加结束符

CONV_DONE:
```

```
    POP DI
    POP DX
    POP CX
    POP BX
    POP AX
    RET
NUM_TO_STR ENDP
```

2. 数字转十进制字符串（带符号）

```
; 输入：AX = 有符号数字
; 输出：DI = 缓冲区地址

NUM_TO_STR_SIGNED PROC
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH DI

    ; 检查是否为负
    TEST AX, AX
    JNS POSITIVE2
    MOV BYTE PTR [DI], '-'
    INC DI
    NEG AX              ; 取绝对值

POSITIVE2:
    CALL NUM_TO_STR     ; 调用无符号转换

    POP DI
    POP DX
    POP CX
    POP BX
    POP AX
    RET
NUM_TO_STR_SIGNED ENDP
```

3. 数字转十六进制字符串

```
; 输入：AX = 数字
; 输出：DI = 缓冲区地址（"XXXX"形式）

NUM_TO_HEX_STR PROC
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH DI

    MOV CX, 4           ; 4位十六进制数
    MOV BX, AX          ; 保存原值

HEX_CONV_LOOP:
```

```
    ROL BX, 4           ; 循环左移4位
    MOV AX, BX
    AND AL, 0FH         ; 取低4位

    CMP AL, 9
    JBE IS_DIGIT2
    ADD AL, 'A' - 10    ; A-F
    JMP STORE_CHAR

IS_DIGIT2:
    ADD AL, '0'         ; 0-9

STORE_CHAR:
    MOV [DI], AL
    INC DI
    LOOP HEX_CONV_LOOP

    MOV BYTE PTR [DI], '$'  ; 结束符

    POP DI
    POP DX
    POP CX
    POP BX
    POP AX
    RET
NUM_TO_HEX_STR ENDP
```

四、完整的数字输入输出程序

```
; ================================================
; 完整的数字计算器示例
; ================================================
.MODEL SMALL
.STACK 100H

.DATA
    MSG1    DB 'Enter first number: $'
    MSG2    DB 'Enter second number: $'
    MSG3    DB 'Sum: $'
    MSG4    DB 'Difference: $'
    MSG5    DB 'Product: $'
    MSG6    DB 'Quotient: $'
    MSG7    DB 'Remainder: $'

    BUF1    DB 7, ?, 7 DUP('$')  ; 输入缓冲区
    BUF2    DB 7, ?, 7 DUP('$')
    RES_BUF DB 8 DUP('$')        ; 结果缓冲区

    NUM1    DW ?
    NUM2    DW ?
    RESULT  DW ?

.CODE
MAIN PROC
    MOV AX, @DATA
```

```
    MOV DS, AX

    ; 获取第一个数字
    CALL GET_NUMBER
    MOV NUM1, AX

    ; 获取第二个数字
    CALL GET_NUMBER
    MOV NUM2, AX

    ; 显示结果
    CALL NEW_LINE

    ; 加法
    MOV AH, 09H
    LEA DX, MSG3
    INT 21H

    MOV AX, NUM1
    ADD AX, NUM2
    CALL DISPLAY_NUMBER

    ; 减法
    CALL NEW_LINE
    MOV AH, 09H
    LEA DX, MSG4
    INT 21H

    MOV AX, NUM1
    SUB AX, NUM2
    CALL DISPLAY_NUMBER

    ; 乘法
    CALL NEW_LINE
    MOV AH, 09H
    LEA DX, MSG5
    INT 21H

    MOV AX, NUM1
    IMUL NUM2
    CALL DISPLAY_NUMBER_DXAX  ; 32位结果

    ; 除法
    CALL NEW_LINE
    MOV AH, 09H
    LEA DX, MSG6
    INT 21H

    MOV AX, NUM1
    CWD                       ; 扩展为DX:AX
    IDIV NUM2
    PUSH DX                   ; 保存余数

    CALL DISPLAY_NUMBER       ; 显示商

    ; 显示余数
```

```
    MOV AH, 09H
    LEA DX, MSG7
    INT 21H

    POP AX                  ; 恢复余数
    CALL DISPLAY_NUMBER

    ; 退出
    MOV AH, 4CH
    INT 21H
MAIN ENDP

; ================================================
; 获取用户输入的数字
; 返回：AX = 数字
; ================================================
GET_NUMBER PROC
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH SI

    ; 显示提示
    CALL NEW_LINE
    MOV AH, 09H
    LEA DX, MSG1
    INT 21H

    ; 输入字符串
    MOV AH, 0AH
    LEA DX, BUF1
    INT 21H

    ; 转换为数字
    LEA SI, BUF1+2
    MOV CL, BUF1+1          ; 实际长度
    MOV CH, 0

    ; 在字符串末尾添加结束标记
    MOV BX, SI
    ADD BX, CX
    MOV BYTE PTR [BX], '$'

    CALL STR_TO_NUM_SIGNED

    POP SI
    POP DX
    POP CX
    POP BX
    RET
GET_NUMBER ENDP

; ================================================
; 显示数字（16位有符号）
; 输入：AX = 数字
; ================================================
```

```
DISPLAY_NUMBER PROC
    PUSH AX
    PUSH DI

    LEA DI, RES_BUF
    CALL NUM_TO_STR_SIGNED

    MOV AH, 09H
    LEA DX, RES_BUF
    INT 21H

    POP DI
    POP AX
    RET
DISPLAY_NUMBER ENDP

; ================================================
; 显示双字数字（32位有符号）
; 输入：DX:AX = 数字
; ================================================
DISPLAY_NUMBER_DXAX PROC
    PUSH AX
    PUSH DX
    PUSH DI

    ; 简单处理：只显示低16位（对于大数字需要完整实现）
    LEA DI, RES_BUF
    CALL NUM_TO_STR_SIGNED

    MOV AH, 09H
    LEA DX, RES_BUF
    INT 21H

    POP DI
    POP DX
    POP AX
    RET
DISPLAY_NUMBER_DXAX ENDP

; ================================================
; 字符串转数字（有符号）
; 输入：SI = 字符串地址
; 输出：AX = 数字
; ================================================
STR_TO_NUM_SIGNED PROC
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH SI

    XOR AX, AX
    XOR CX, CX          ; CX=0:正数，CX=1:负数
    MOV BX, 10

    ; 检查符号
    CMP BYTE PTR [SI], '-'
```

```
    JNE CHECK_PLUS
    MOV CX, 1
    INC SI
    JMP CONV_LOOP3

CHECK_PLUS:
    CMP BYTE PTR [SI], '+'
    JNE CONV_LOOP3
    INC SI              ; 跳过正号

CONV_LOOP3:
    MOV DL, [SI]
    CMP DL, '$'
    JE CONV_DONE3
    CMP DL, '0'
    JB CONV_DONE3
    CMP DL, '9'
    JA CONV_DONE3

    SUB DL, '0'
    PUSH DX
    MUL BX
    POP DX
    ADD AX, DX

    INC SI
    JMP CONV_LOOP3

CONV_DONE3:
    ; 处理符号
    TEST CX, CX
    JZ POSITIVE3
    NEG AX

POSITIVE3:
    POP SI
    POP DX
    POP CX
    POP BX
    RET
STR_TO_NUM_SIGNED ENDP

; ================================================
; 数字转字符串（有符号）
; 输入：AX = 数字，DI = 缓冲区地址
; ================================================
NUM_TO_STR_SIGNED PROC
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH DI

    ; 检查是否为负
    TEST AX, AX
    JNS NOT_NEG
```

```
    MOV BYTE PTR [DI], '-'
    INC DI
    NEG AX

NOT_NEG:
    MOV CX, 0
    MOV BX, 10

    ; 处理0
    TEST AX, AX
    JNZ DIVIDE3
    MOV BYTE PTR [DI], '0'
    INC DI
    JMP FINISH3

DIVIDE3:
    XOR DX, DX
    DIV BX
    PUSH DX
    INC CX
    TEST AX, AX
    JNZ DIVIDE3

POP_LOOP3:
    POP DX
    ADD DL, '0'
    MOV [DI], DL
    INC DI
    LOOP POP_LOOP3

FINISH3:
    MOV BYTE PTR [DI], '$'

    POP DI
    POP DX
    POP CX
    POP BX
    POP AX
    RET
NUM_TO_STR_SIGNED ENDP

; ================================================
; 工具函数
; ================================================
NEW_LINE PROC
    PUSH AX
    PUSH DX

    MOV AH, 02H
    MOV DL, 0DH
    INT 21H
    MOV DL, 0AH
    INT 21H

    POP DX
    POP AX
```

```
    RET
NEW_LINE ENDP

END MAIN
```

五、常用的数字处理宏定义

```
; ===============================================
; 数字输入输出宏库
; ===============================================

; 宏：读取十进制数字到AX
READ_DECIMAL MACRO
    LOCAL READ_LOOP, DONE_READ
    PUSH BX
    PUSH CX
    PUSH DX

    XOR AX, AX
    XOR CX, CX          ; 符号标志
    MOV BX, 10

    ; 读取第一个字符（可能是符号）
    MOV AH, 01H
    INT 21H

    CMP AL, '-'
    JE NEGATIVE
    CMP AL, '+'
    JE POSITIVE_CHAR
    JMP CHECK_DIGIT

NEGATIVE:
    MOV CX, 1           ; 设置负数标志

POSITIVE_CHAR:
    ; 读取下一个字符
    MOV AH, 01H
    INT 21H

CHECK_DIGIT:
    CMP AL, '0'
    JB DONE_READ
    CMP AL, '9'
    JA DONE_READ

    SUB AL, '0'
    PUSH AX
    MOV AX, BX
    MUL BX              ; AX × 10
    POP DX
    ADD AX, DX

    ; 继续读取
    MOV AH, 01H
```

```
    INT 21H
    JMP CHECK_DIGIT

DONE_READ:
    ; 处理符号
    TEST CX, CX
    JZ READ_END
    NEG AX

READ_END:
    POP DX
    POP CX
    POP BX
ENDM

; 宏：显示AX中的十进制数字
DISPLAY_DECIMAL MACRO
    LOCAL DISPLAY_LOOP, DISPLAY_DIGIT
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX

    ; 检查是否为负
    TEST AX, AX
    JNS POSITIVE_DISP
    PUSH AX
    MOV DL, '-'
    MOV AH, 02H
    INT 21H
    POP AX
    NEG AX

POSITIVE_DISP:
    MOV CX, 0
    MOV BX, 10

    ; 处理0
    TEST AX, AX
    JNZ DIVIDE_LOOP2
    MOV DL, '0'
    MOV AH, 02H
    INT 21H
    JMP DISPLAY_END

DIVIDE_LOOP2:
    XOR DX, DX
    DIV BX
    PUSH DX
    INC CX
    TEST AX, AX
    JNZ DIVIDE_LOOP2

DISPLAY_LOOP:
    POP DX
    ADD DL, '0'
```

```
    MOV AH, 02H
    INT 21H
    LOOP DISPLAY_LOOP

DISPLAY_END:
    POP DX
    POP CX
    POP BX
    POP AX
ENDM

; 使用示例
.DATA
    NUM DW ?

.CODE
    ; 读取数字
    READ_DECIMAL
    MOV NUM, AX

    ; 显示数字
    MOV AX, NUM
    DISPLAY_DECIMAL
```

六、总结

1. **DOS没有直接的数字I/O**，必须自己实现转换
2. **核心算法**：
    - 输入：`result = result × 10 + digit`
    - 输出：反复除10，余数转为ASCII
3. **处理负数**：需要检查'-'符号
4. **错误处理**：应验证输入是否为有效数字
5. **大数字**：需要32位或64位处理