X86复习

复杂程序解读

```
STACK       SEGMENT PARA STACK
STACK_AREA  DW 100H DUP(?)
STACK_TOP   EQU $-STACK_AREA
STACK       ENDS

DATA        SEGMENT PARA
STRING1     DB 'MY NAME IS DENGTAO', 00H
LEN         EQU 20H
NUM_BUF     DB 3
            DB ?
            DB 4 DUP(?)
; 缓冲区定义
IN_BUF      DB LEN-1
            DB ?
STRING2     DB LEN DUP(?)
; 定义跳转表
CALL_TABLE  DW FUNC0, FUNC1, FUNC2, FUNC3, FUNC4, FUNC5
DATA        ENDS

CODE        SEGMENT
            ASSUME CS:CODE, DS:DATA
            ASSUME SS:STACK

MAIN        PROC
            ; 内存不能到段寄存器，要通用寄存器中转
            MOV AX, STACK
            MOV SS, AX
            MOV SP, STACK_TOP
            MOV AX, DATA
            MOV DS, AX
            ; 段寄存器不能到段寄存器
            PUSH DS
            POP ES

            ; 主循环
MAIN_LP:    MOV SI, OFFSET NUM_BUF ;通过堆栈传递参数
            PUSH SI
            ; 调用读数字函数，结果放在AX中
            CALL READ_NUM
```

```
            ; 左移计算从跳转地址
            SHL AX, 1
            MOV BX, OFFSET CALL_TABLE
            ADD BX, AX
            ; 跳转到BX对应DS段中值
            ; CALL_TABLE编译完成后为一个数组，保存了每个标签相对CS段偏移
            ; 不能JMP BX !!!!
            JMP [BX]

            ; 跳出主循环
FUNC0:      JMP MAIN_END

            ; 读字符串
FUNC1:      MOV SI, OFFSET IN_BUF
            PUSH SI
            CALL READ_STR
            JMP MAIN_LP

            ; 查找字符出现次数
FUNC2:      MOV SI, OFFSET STRING2
            PUSH SI
            XOR DX, DX
            MOV DL, 'A'
            PUSH DX
            CALL FIND
            JMP MAIN_LP

            ; 比较两个字符串
FUNC3:      MOV SI, OFFSET STRING1
            PUSH SI
            MOV SI, OFFSET STRING2
            PUSH SI
            CALL STR_CMP
            JMP MAIN_LP

            ; 将字符串1拷贝到2位置
FUNC4:      MOV SI, OFFSET STRING1
            PUSH SI
            MOV DI, OFFSET STRING2
            PUSH DI
            CALL MEMMOVE
            JMP MAIN_LP

            ; 显示字符串
FUNC5:      MOV SI, OFFSET STRING2
            PUSH SI
            CALL DISPLAY_STR
            JMP MAIN_LP

            ; 结束程序
MAIN_END:   MOV AH, 4CH
            INT 21H
MAIN        ENDP
READ_NUM    PROC
            ; 堆栈传参标准格式
            ; 如果PROC FAR则BP+6
```

```
            PUSH BP
            MOV BP, SP
            MOV DX, [BP+4]

            MOV AH, 0AH
            INT 21H

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

            MOV SI, [BP+4]
            MOV DL, BYTE PTR [SI+2]
            ; '0' == 30H
            SUB DL, '0'
            XOR AX, AX
            MOV AL, DL
            ; 记得还原BP
            POP BP
            ; RET 2n，n为参数数量
            RET 2
READ_NUM    ENDP

READ_STR    PROC
            PUSH BP
            MOV BP, SP
            MOV DX, [BP+4]

            MOV AH, 0AH
            INT 21H

            MOV DL, 0AH
            MOV AH, 02H
            INT 21H
            POP BP
            RET 2
READ_STR    ENDP

FIND        PROC
            PUSH BP
            MOV BP, SP
            MOV DX, [BP+4]
            MOV SI, [BP+6]
            XOR AX, AX

FIND_LP1:   CMP DL, BYTE PTR DS:[SI]
            JZ PULS
            JMP EL
PULS:       INC AX
EL:         CMP BYTE PTR DS:[SI], 00H
            JZ FIND_OUT
            INC SI
            JMP FIND_LP1

FIND_OUT:   MOV DL, AL
            ADD DL, '0'
```

```
            MOV AH, 02H
            INT 21H

            MOV DL, 0AH
            MOV AH, 02H
            INT 21H
            POP BP
            RET 4
FIND        ENDP


STR_CMP     PROC
            PUSH BP
            MOV BP, SP
            MOV SI, [BP+4]
            MOV DI, [BP+6]
            CLD
CMP_LP1:    LODSB
            CMP AL, ES:[DI]
            JA CMP_L1
            JB CMP_L2
            CMP AL, 00H
            JZ CMP_L3
            INC DI
            JMP SHORT CMP_LP1

CMP_L1:     MOV DL, '1'
            MOV AH, 02H
            INT 21H
            JMP SHORT RETURN

CMP_L2:     MOV DL, '-'
            MOV AH, 02H
            INT 21H
            MOV DL, '1'
            INT 21H
            JMP SHORT RETURN

CMP_L3:     MOV DL, '0'
            MOV AH, 02H
            INT 21H

RETURN:
            MOV DL, 0AH
            MOV AH, 02H
            INT 21H
            POP BP
            RET 4

STR_CMP     ENDP

MEMMOVE     PROC
            PUSH BP
            MOV BP, SP
            MOV DI, [BP+4]
            MOV SI, [BP+6]
```

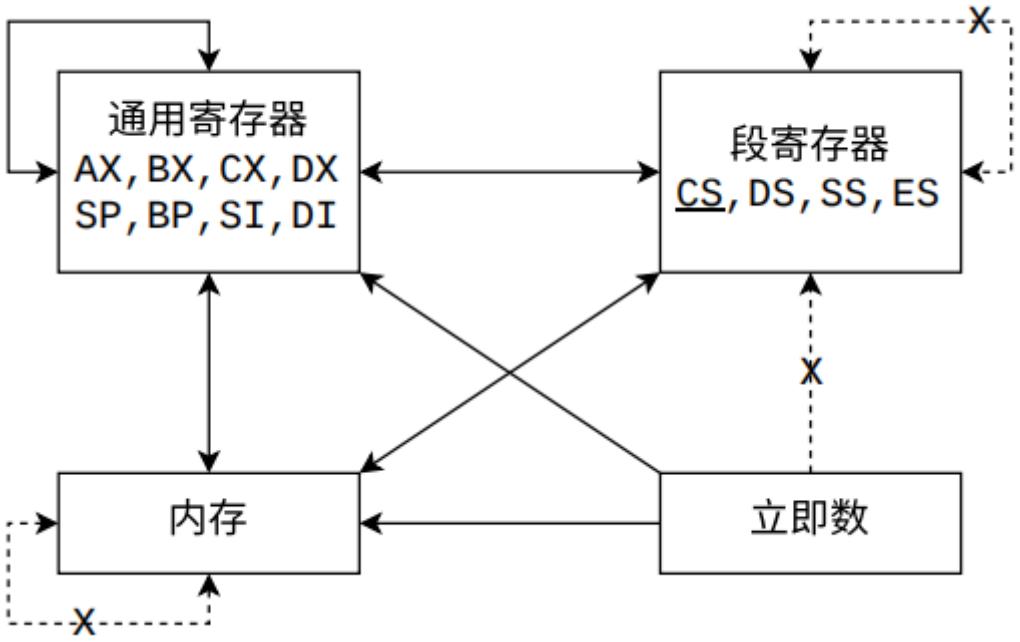
```
            XOR CX, CX
            PUSH SI
            CLD
MEM_LP1:    LODSB
            INC CX
            CMP AL, 00H
            JNZ MEM_LP1

            POP SI
            DEC CX
            CLD
MEM_LP2:    MOVSB
            LOOP MEM_LP2
            MOV BYTE PTR [DI], 00H

            POP BP
            RET 4

MEMMOVE     ENDP

DISPLAY_STR PROC
            PUSH BP
            MOV BP, SP
            MOV SI, [BP+4]
DIS_LP1:    MOV DL, BYTE PTR[SI]
            CMP DL, 00H
            JZ OUTLP1
            INC SI
            MOV AH, 02H
            INT 21H
            JMP DIS_LP1
OUTLP1:     MOV DL, 0AH
            MOV AH, 02H
            INT 21H
            POP BP
            RET 2

DISPLAY_STR ENDP

CODE        ENDS
            END MAIN
```

牢记

一、系统调用 INT 21H

- `MOV AH, 4CH` 结束程序返回DOS
- `MOV AH, 02H` 表示显示字符，字符放在DL中
- `MOV AH, 09H` 表示显示字符串，字符串地址放在DX中
- `MOV AH, 0AH` 读入字符串，缓存区地址放在DX中
- `MOV AH, 01H` 读入一个字符，字符放在AL中

二、数据通路



注意: 使用段寄存器作为目的操作数时，不允许用 CS （修改 CS 只能通过段间跳转指令）。

三、寻址方式

- **立即寻址**: 指令操作数包含在指令中，为一个常量或常数
  - MOV AX, 1
  - MOV AX, X ;X EQU 100
- **寄存器寻址**: 指令操作数为 CPU 的寄存器
  - MOV AX, BX
- **直接寻址**: 操作数偏移地址 EA 在指令中给出，如变量名
  - MOV AX, [100H]
  - MOV AX, X ;X DW 100H
- **寄存器间接寻址**: 操作数地址 EA 位于间指寄存器 (BX , BP , SI , DI) 中
  - MOV AX, [SI] ;DS段
  - MOV AX, [BP] ;SS段
- **寄存器相对寻址**: 操作数地址 EA 由间指寄存器 + 8 位或 16 位的常量组成
  - MOV AX, [SI+100H]
- **基址变址寻址**: 操作数地址 EA 为一个基址寄存器和一个变址寄存器之和
  - MOV AX, [BX+SI]
  - MOV AX, [BX+SI+100H]

四、跳转方式

- **段内直接寻址**
  - JMP l1
- **段内间接寻址**
  - MOV AX, OFFSET p1
  - CALL AX
  - CALL [BX]
- **段间直接寻址**
  - CALL FAR PTR p2
- **段间间接寻址**
  - JMP DWORD PTR [BX+INTERS]

五、常数

美国信息交换标准码

‘A’—’Z’　　41h — 5Ah　　‘0’—’9’　　31h — 39h

‘a’—’z’　　61h — 7Ah　　空格—20h　‘\0’ — 00h

换行—0Ah　　回车 — 0Dh　　换页 — 0Ch　文件尾 — 1Ah

DOS(Windows)文本换行——0dh,0ah;　Unix文本换行——0ah