·注意进入宏定义和函数，ax，bx等的pop和push

·注意输出后ah的清空，或输出前ah是否被占用

·注意数组是db还是dw，以及指针读取问题

·注意si和di和含x寄存器都是16位

·注意loop语句先对cx-1再判断cx=0

·XCHG ax,bx 作用是ax和bx内容交换

·字节无符号数相乘:(SRC)\*(AL) → AX;

字无符号数相乘 :(SRC)\*(AX) → DX: AX;

·字节除法:(AX)/(SRC) → AL, 余数→ AH

字除法:(DX:AX)/(SRC) → AX, 余数→ DX

·指令格式:CBW/CWD;隐含使用AH/DX

CBW :扩展AL中的符号位至AH中

CWD:扩展AX中的符号位至DX中

·jz 为零跳转 zf=1

jnz 非零跳转 zf=0

jc 有进位则跳转 cf=1

jnc 无进位则跳转 cf=0

jo 溢出则跳转 of=1

jno 无溢出则跳转 of=0

js 如果符号位为1则跳转 sf=1

jns 如果符号位为零则跳转 sf=0

jp 如果奇偶标志位为1（奇偶性为偶）则跳转 pf=1

jnp 如果奇偶标志位为0（奇偶性为奇）则跳转 pf=0

无符号用ja jb

Ja 多于则跳转

Jae jbe

Jb 少于则跳转

Je 相等则跳转

Jne不相等则跳转

带符号用jg jl

Jge 大于等于

Jg 大于

Jl 小于

Jle 小于等于

|  |
| --- |
| ;定义换行宏定义  enterline macro  push dx  push ax  mov dl,13  mov ah,2  int 21h  mov dl,10  mov ah,2  int 21h  pop ax  pop dx  endm  ;定义输出字符串的指令  putstr macro string  push dx  push ax  lea dx, string  mov ah, 09h  int 21h  pop ax  pop dx  endm  字符串定义和输入  buf db 21,?,21 dup(‘$’)  lea dx, buf  xor dh, dh  mov ah, 10  int 21h  enterline |

|  |
| --- |
| ;数据定义  buf db 100, ?, 100 dup(0)  arr db 100h dup(0)  ;堆栈区定义  db 128 dup(0) |

|  |
| --- |
| 数组排序程序定义变量  arr db 100h dup(0)  arrlen dw 0  arrpointer dw 0  is\_neg dw 0  strlen dw 0  string\_neg db '-$'  string\_space db ' $'  a2n\_trans\_flag dw 0  temp db 0  ltemp db 0  sort\_flag dw 0 |
| a2n proc  a2n\_begin:;buf+1  xor di, di  lea di, arr  mov arrpointer, di存数组[0]的地址  mov cl,buf+1 ;将实际接收字符数送入cl  xor ch,ch ;清零语句,用异或  xor di,di  xor dx,dx  mov strlen, cx  mov bx,1 ; 放置权重，初值是1  lea si,buf+2  add si,cx  dec si  ;从后向前循环检查输入字符并转换成数字  a2n\_cov:mov al,[si]  cmp al, 20h  je a2n\_skip\_space  cmp al, '-'  je a2n\_neg\_num  cmp al,'0'  jb a2n\_error  cmp al,'9'  ja a2n\_error    sub al,30h ;转换成数字  xor ah,ah ;清零  mul bx ;从个位开始，乘以权重  cmp dx,0 ;dx符号位，溢出检测  jne a2n\_error    add di,ax  ;jc:有进位则跳转  jc a2n\_error    mov ax,bx  mov bx,10  mul bx ; 调整权重，各位是1，每循环一次乘以10  mov bx,ax    dec si  loop a2n\_cov ; 每循环一次，处理一位  a2n\_neg\_out:  ;保存数据  mov ax,di ; 将结果保存  cmp is\_neg, 1  jne a2n\_pos2  neg ax  a2n\_pos2:mov di, arrpointer; 指针保存  mov [di], al ;结果存入数组  inc arrpointer ; 指针推进  inc arrlen  mov is\_neg, 0  ret  a2n\_error: lea dx,err  mov ah,9  mov is\_neg, 0  mov arrlen, 0  mov arrpointer, 0  int 21h  enterline  jmp begin;;;;;;;;;输入字符串位置;;;;;;;;;;  a2n\_skip\_space:cmp strlen, cx  je a2n\_error  dec si  dec cl  mov al, [si]  cmp al, '-'  je a2n\_error  mov al, [si]  cmp al, ' '  je a2n\_error    ;保存数据  mov ax,di ; 将结果保存  cmp is\_neg, 1  jne a2n\_pos1  neg ax  a2n\_pos1:mov di, arrpointer; 指针保存  mov [di], al ;结果存入数组  inc arrpointer ; 指针推进  inc arrlen  mov is\_neg, 0  xor di, di  mov bx, 1  xor dx, dx  jmp a2n\_cov    a2n\_neg\_num:cmp strlen, cx  je a2n\_error  dec si  dec cl  mov al, [si]  cmp al, '-'  je a2n\_error  mov is\_neg, 1  cmp cx, 0  je a2n\_neg\_out  mov al, [si]  cmp al, ' '  jne a2n\_error  jmp a2n\_cov  a2n endp |
| traverse proc  xor ax, ax  xor bx, bx  xor cx, cx  xor dx, dx  xor si, si  xor di, di  mov cx, arrlen  mov si, arrpointer ;pointer此时是数组最大index+1  dec si;所以需要-1  traverse\_tloop:  mov al, [si]  xor ah, ah  cmp al, 0h  jge traverse\_tln  putstr string\_neg  neg al  traverse\_tln:  Push cx  call num2asc  Pop cx  dec si  loop traverse\_tloop  ret  traverse endp |
| ;需要ax提前存储好待转换的数，再调用此函数，此函数转换后直接输出数字+空格  ; n2a\_trans\_flag dw 0  ; string\_neg db ‘-’  ; string\_space db ‘ ‘  n2a proc  xor cx, cx  xor bx, bx  xor dx, dx  mov bx,10000  n2a\_cov\_back:  xor dx,dx ;清零  div bx ;ax是被除数，bx是除数，ax是商，dx是余数  mov cx,dx ;余数先存放在cx  cmp n2a\_trans\_flag,0  jne n2a\_nor1  cmp ax,0 ;商是零  je n2a\_cont  n2a\_nor1:  mov dl,al  add dl,30h ;转换成ASCII码输出  mov ah,2 ;输出  int 21h  mov n2a\_trans\_flag,1    n2a\_cont:  cmp bx,10  je n2a\_outer    xor dx,dx ;清零  mov ax,bx ;递减权重  mov bx,10  div bx  mov bx,ax    mov ax,cx;将余数送到被除数中继续除  jmp n2a\_cov\_back    n2a\_outer:  mov dl,cl;个位输出  add dl,30h  mov ah,2  int 21h  putstr string\_space  mov n2a\_trans\_flag, 0  xor di, di  ret  n2a endp |
| 进制转换:改变上方红字即可  to d:上方  to h: mov bx,4096 cmp bx,16 mov bx,10  to b: mov bx,10000b cmp bx,10b mov bx,10b  注意to h的outer需要变为如下  mov dl,cl;个位输出  ;如果是0-9的数字  add dl,30h  cmp dl,39h  jle d2h\_continue  add dl, 27h;字母小写  ;add dl, 7h;字母大写  d2h\_continue:  mov ah,2  int 21h  enterline  mpop  ret |
| 冒泡  bubble proc  xor di, di  xor si, si  xor cx, cx  xor bx, bx  xor ax, ax  ;外层循环次数  mov cx, arrlen  oloop:  mov ltemp, cl  ;存储外层循环次数  push cx  ;外层数组指针  lea si, arr  ;获取内层循环次数  ;dec ltemp  mov cl, ltemp  ;内层数组指针  mov di, si  inc di  iloop:  ;由于loop先自减再判断  ;所以cx=1的时候，实际上循环就已经该停止了  cmp cl ,1  je inner  ;比大小，由于数组倒插入，所以要求大到小排序，这里需要小到大排序  mov al, [si]  xor ah, ah  mov bl, [di]  xor bh, bh  cmp al, bl  jle continue  ;交换  mov temp, al  mov al, bl  mov bl, temp  mov [si], al  mov [di], bl  continue:  inc di;内层循环指针推进  inc si  inner: loop iloop  pop cx  loop oloop  ret  bubble endp |

|  |
| --- |
| 大小写字母转换  b2s proc  mpush  mov cl, buf+1  xor ch, ch  lea si, buf+2  b2s\_loop:  mov al, [si]  cmp al,'A'  jb b2s\_no  cmp al,'Z'  ja b2s\_no  add al, 20h  mov [si], al  b2s\_no:  inc si  loop b2s\_loop  mpop  ret  b2s endp  如果是小写，只需改变loop这里为  mov al, [si]  cmp al,'a'  jb s2b\_no  cmp al,'z'  ja s2b\_no  sub al, 20h  mov [si], al |

|  |
| --- |
| 作业5：统计字符串  数据区cc dw 0  sc dw 0  nc dw 0  ec dw 0  buf db 30, ?, 30 dup(0)  string1 db 'Please enter a line of string:$'  string2 db 'A-Z and a-z :$'  string3 db 'Space :$'  string4 db 'Numbers :$'  string5 db 'Other chars :$'  代码区  putstr string1    lea dx, buf  mov ah,10  int 21h  enterline    lea si, buf+2  mov cl, buf+1  xor ch, ch  forward:  mov bl, [si]  ;判断空格  cmp bl, ' '  je SPACE  ;判断数字  cmp bl, 30h  jb not\_num  cmp bl, 39h  ja not\_num  jmp NUM  not\_num:  ;判断字母  cmp bl, 41h  jb not\_char  cmp bl, 7ah  ja not\_char  ;到这里能判断字符在A-z之间  cmp bl, 5ah  jle CHAR ;大写字母  cmp bl, 61h  jge CHAR ;小写字母  not\_char:  ;判断回车  cmp bl, 13h  je ot  inc ec  ot:  inc si  loop forward    ;打印结果  putstr string2  mov ax, cc  call num2asc  enterline    putstr string3  mov ax, sc  call num2asc  enterline    putstr string4  mov ax, nc  call num2asc  enterline    putstr string5  mov ax, ec  call num2asc  enterline    jmp stop    SPACE:  inc sc  jmp ot  NUM:  inc nc  jmp ot  CHAR:  inc cc  jmp ot |