



Main /  
Temi d'esame

- [tde1](#) \*soluzione
- [tde2](#) \*soluzione
- [tde3](#) \*soluzione
- [tde4](#) \*soluzione
- [tde5](#) \*soluzione
- [tde6](#) \*soluzione

Soluzioni a cura del [group6](#)

Soluzione Tema d'Esame 1

forum?

wiki  
people

2009

pmwiki.org

Cookbook (addons)  
Skins (themes)  
PITS (issue tracking)  
Mailing Lists

edit SideBar

<pre>tde1.ASM  public _stazione .model small .stack .data  cont_ril dw 0 ; inizializzazione contatore rilevamenti max      dw 0 ; valore di appoggio per massimo .code  _stazione proc      push bp     mov bp,sp     push si     push di     push bx     push cx     push dx      xor bx,bx ; inizializzazione contatore stazioni     mov di,[bp+8] ; indirizzo dati_output  stazioni:     mov si,[bp+6] ; indirizzo dati_input     mov cx,[bp+4] ; int n, numero rilevamenti     xor ax,ax ; inizializzazione contatore temperature     mov cont_ril,0 ; inizializzazione contatore rilevamenti  rilevamenti:     cmp bx,[si] ; controllo se stazione corrente     jne continua ; no --&gt; salta rilevazione     add ax,[si+2] ; si --&gt; prendi la temperatura e somma a quelle precedenti     inc cont_ril ; incremento contatore rilevamenti per stazione     push ax ; salvo ax nello stack     mov ax,[si+2] ; metto in ax la temperatura corrente     cmp ax,max ; ax &gt; max?     jl continua_pop ; no --&gt; continua     mov max,ax ; si --&gt; max = ax  continua_pop:     pop ax ; ripristino il valore di ax  continua:     add si,4 ; si muove alla coppia successiva     loop rilevamenti      cwd ; inizializzazione per divisione tra word     idiv cont_ril ; esegue la media     mov [di],ax ; salva il risultato nel vettore di output     add di,2 ; sposta il puntatore del vettore di output     inc bx ; incremento stazione      cmp bx,4 ; controllo se stazioni esaurite     jb stazioni ; no --&gt; continua a scorrere      mov ax,max ; valore di ritorno      pop dx     pop cx     pop bx     pop di     pop si     pop bp      ret _stazione endp end</pre>	<pre>tde1.C  #include &lt;stdio.h&gt; extern int stazione (int n, int *dati_input, int *dati_output); void main() {     int n = 6;     int dati_input[18] = {1,0,1,-1,3,3,0,1,2,4,0,1,2,6,3,3,1,-2};     int dati_output[4];     int max=stazione(n, dati_input, dati_output);     printf("Stringa di input: ");     for (int i=0; i&lt;18; i++)         printf("%d",dati_input[i]);      printf("\nStringa di output: ");     for (int i=0; i&lt;4; i++)         printf("%d",dati_output[i]);      printf("\nTemperatura massima rilevata: %d",max); }</pre>
--	---

Soluzione Tema d'Esame 2

<pre>tde2.ASM  public _compatta .model small .stack .data  byte_tot dw 0 ; contatore byte min      dw 0FFFFh ; valore minimo spazio libero offset_min dw 0 ; valore indirizzo minimo spazio libero ultimo   dw 0 ; spostamento ultimo blocco dim_blocco dw 0 ; dimensione ultimo blocco sostituito db 0 ; flag che indica se il blocco e' gia' stato sostituito  .code  _compatta proc     push bp     mov bp,sp     push si     push di     push bx     push cx     push dx      mov di,[bp+4] ; primo parametro: *tabella     mov bx,[bp+6] ; secondo parametro: n      dec bx ; calcolo spostamento per prendere l'ultimo blocco     shl bx,2 ; 4*(n-1)     mov ultimo,bx      add bx,2 ; spostamento per arrivare a ultimo elemento     mov ax,[di+bx] ; ultimo elemento     sub bx,2 ; spostamento per arrivare a penultimo elemento     sub ax,[di+bx] ; calcolo dimensione blocco     mov dim_blocco,ax;salvo il valore nella variabile d'appoggio      mov cx,[bp+6] ; n  blocco:     mov ax,[di+2] ; secondo elemento blocco     sub ax,[di]</pre>	<pre>tde2.C  #include &lt;stdio.h&gt; extern int compatta (int *tabella, int n); void main() {     int n = 5;     int tabella[10] = {10,20,70,90,130,200,1000,1500,2000,2030};     printf("Tabella input: \n");     for (int i=0; i&lt;10; i++)         printf("%d\n",tabella[i]);      int byte_totali=compatta(tabella,n);     printf("\nTabella output: ");     for (int i=0; i&lt;4; i++)         printf("%d",tabella[i]);      printf("\nNumero di byte occupati: %d",byte_totali); }</pre>
---	--

```

        add byte_tot,ax ; sommo il valore nella variabile di appoggio
        cmp cx,1       ; ultimo elemento non devo calcolare spazio libero
        je end_blocco
        add di,2       ; secondo elemento blocco
        mov bx,[di+2]  ; primo elemento blocco successivo
        sub bx,[di]    ; calcolo spazio libero tra i 2 blocchi adiacenti
        add di,2       ; primo elemento blocco successivo
        cmp dim_blocco,bx; spazio libero >= dimensione ultimo blocco?
        ja continua   ; no --> continua
        cmp min,bx    ; spazio libero < min?
        jbe continua  ; no --> continua
        mov min,bx    ; nuovo minimo
        mov offset_min,di; offset primo elemento blocco da sostituire

continua:
        loop blocco

end_blocco:
        cmp offset_min,0; nessuno spazio libero sufficiente
        je fine

        mov di,[bp+4] ; reset blocco iniziale
        mov cx,[bp+6] ; inizializzazione contatore a n
        inserimento:
        cmp cx,1      ; ultimo blocco?
        je ult_blocco ; si --> ult_blocco
        cmp sostituito,1 ; il blocco e' stato sostituito?
        je sposta     ; si --> sposta
        cmp offset_min,di ; offset blocco da sostituire?
        jne scorri    ; no --> scorri

        mov ax,[di-2] ; secondo elemento blocco precedente
        push [di+2]   ; salvo nello stack secondo e
        push [di]     ; primo elemento del blocco da sostituire
        mov [di],ax   ; sostituisco primo elemento
        add ax,dim_blocco ; calcolo secondo elemento
        mov [di+2],ax ; sostituisco secondo elemento
        mov sostituito,1 ; il blocco e' stato sostituito
        jmp scorri

        sposta:
        pop ax        ; primo elemento blocco
        pop bx        ; secondo elemento blocco
        push [di+2]   ; salvo nello stack secondo e
        push [di]     ; primo elemento del blocco da sostituire
        mov [di],ax   ; primo elemento blocco da scorrere
        mov [di+2],bx ; secondo elemento blocco da scorrere
        jmp scorri

        ult_blocco:
        pop ax        ; primo elemento blocco
        pop bx        ; secondo elemento blocco
        mov [di],ax   ; primo elemento blocco da scorrere
        mov [di+2],bx ; secondo elemento blocco da scorrere
        jmp fine

        scorri:
        add di,4      ; spiazzamento blocco successivo
        loop inserimento

        fine:
        mov ax,byte_tot ; valore di ritorno

        pop dx
        pop cx
        pop bx
        pop di
        pop si
        pop bp

        ret

_compatta endp
end

```

### Soluzione Tema d'Esame 3

#### tde3.ASM

```

public _vecchio
.model small
.stack
.data

giorno_n db 0 ;giorno nascita
mese_n db 0 ;mese nascita
anno_n dw 0 ;anno nascita

giorno_m db 0 ;giorno morte
mese_m db 0 ;mese morte
anno_m dw 0 ;anno morte

vita dw 0 ;eta' in giorni

giorni db 0,31,28,31,30,31,30,31,31,30,31,31,31 ;vettore con i giorni dei mesi
                                           ;vett+1 = gennaio
                                           ;vett+2 = febbraio
                                           ;...

n_max dw 0 ;persona che ha vissuto piu' a lungo
vita_max dw 0 ;eta' massima tra le persone

.code

_vecchio proc

    push bp
    mov bp,sp
    push bx
    push cx
    push dx
    push si
    push di

    mov si,[bp+6] ; *vett
    mov cx,[bp+4] ; n
    lea di,giorni ; offset vettore giorni mesi

    persona:
    sub sp,6 ; per ogni persona calcolo l'eta' in giorni
    push si ; lascio spazio per giorno, mese e anno
            ; metto si nello stack così la procedura potrà
            ;prenderlo come parametro

    call calcolo_data
    pop si
    pop ax ; valore di ritorno procedura: giorno_nascita
    mov giorno_n,al ; salvo il valore nella variabile
    pop ax ; valore di ritorno procedura: mese_nascita
    mov mese_n,al ; salvo il valore nella variabile
    pop anno_n ; valore di ritorno procedura: anno_nascita

    add si,8 ; spostato si verso data morte
    sub sp,6 ; spazio per i valori di ritorno dalla procedura
            ; per il calcolo della data
    push si ; metto si nello stack così la procedura potrà
            ; prenderlo come parametro

    call calcolo_data
    pop si
    pop ax ; valore di ritorno procedura: giorno_morte
    mov giorno_m,al ; salvo il valore nella variabile
    pop ax ; valore di ritorno procedura: mese_morte

```

#### tde3.C

```

#include <stdio.h>
extern int vecchio (char *vett,int n);
void main()
{
    int n = 3;
    char vett[] = "301119610405200602021894270119752610190229121950";
    int older=vecchio(vett, n);
    printf("Stringa di input: %s",vett);
    printf("\nLa persona vissuta piu' a lungo e': %d",older);
}

```

```

mov mese_m,al      ; salvo il valore nella variabile _
pop anno_m         ; valore di ritorno procedura: anno_nascita

mov al,mese_m
cmp al,mese_n      ; mese_morte > mese_nascita ?
jbe continua_1     ; no --> continua_1
mov ax,anno_m      ; si --> ultimo anno compiuto interamente, posso sommare
                    ; i giorni e i mesi senza rotazione
sub ax,anno_n      ; anno_morte - anno_nascita
mov bx,365         ; giorni in un anno
xor dx,dx          ; azzero dx per la moltiplicazione
mul bx             ; (anno_m - anno_n)*365 --> giorni per gli anni vissuti
mov vita,ax        ; inizio accumulo giorni vissuti
mov bl,mese_n      ; spostamento vettore giorni per avere il numero di giorni
                    ; nel mese specificato
mov bh,0           ; azzero bh perchè mi serve bx completo
mov al,[di+bx]     ; giorni del mese di nascita
sub al,giorno_n    ; giorni mese totale - giorno_nascita
mov ah,0           ; azzero ah perchè mi serve ax completo per sommare la word vita
add vita,ax        ; aggiungi giorni
mov bl,giorno_m    ; basta aggiungere i giorni del mese di morte
mov bh,0           ; azzero bh perchè mi serve bx completo per sommare la word vita
add vita,bx        ; aggiungi giorni data morte
mov bl,mese_n      ; devo calcolare i giorni dei mesi tra mese_nascita e mese_morte
inc bl             ; i giorni del mese di nascita li ho già contati
calcola_giorni_1:  ; calcola quanti giorni mancano per arrivare all'ultimo anno
mov bh,0           ; azzero bh perchè mi serve bx completo
mov al,[di+bx]     ; giorni del mese considerato
mov ah,0           ; azzero ah perchè mi serve ax completo per sommare la word vita
add vita,ax        ; aggiungi giorni mese considerato
inc bl             ; avanza mese
cmp bl,mese_m      ; mese_nascita < mese_morte?
jb calcola_giorni_1 ; si --> continua ciclo
                    ; no --> conteggio terminato
jmp calcola_max    ; vai al calcolo del max

continua_1:
mov al,mese_m
cmp al,mese_n      ; mese_morte = mese_nascita ?
je continua_2      ; si --> continua_2
                    ; no --> mese_morte < mese_nascita
mov ax,anno_m      ; per il calcolo dei giorni devo considerare un anno in meno
sub ax,anno_n      ; rispetto alla differenza tra morte e nascita
dec ax             ; (anno_m - anno_n)-1
mov bx,365         ; giorni in un anno
xor dx,dx          ; azzero dx per la moltiplicazione
mul bx             ; giorni della differenza anni
mov vita,ax        ; inizio a contare i giorni
mov bl,mese_n      ; devo sommare i giorni che mancano alla fine del mese di nascita
mov bh,0           ; azzero bh perchè mi serve bx completo
mov al,[di+bx]     ; giorni totali del mese di nascita
sub al,giorno_n    ; giorni[mese_nascita] - giorno_nascita
mov ah,0           ; azzero ah perchè mi serve ax completo per sommare la word vita
add vita,ax        ; aggiungi giorni
mov bl,giorno_m    ; devo aggiungere i giorni del mese di morte
mov bh,0           ; azzero bh perchè mi serve bx completo per sommare la word vita
add vita,bx        ; aggiunge giorni mese_morte

cmp mese_m,1       ; mese_morte = 1 (--> gennaio)? metto 13 perchè sarebbe il valore
                    ; incrementato dal ciclo per il calcolo dei giorni
jne no_gennaio     ; no --> continua
mov mese_m,13      ; si --> mese_m = 13
no_gennaio:
mov bl,mese_n      ; mese_nascita mese di partenza per calcolo giorni
inc bl             ; incremento di 1 perchè i giorni del mese_nascita sono stati già sommati
calcola_giorni_2:  ; se mese_nascita 13 (gennaio dopo incremento su dicembre)
cmp bl,13          ; no --> continua
jne no_round       ; si --> metto 1 (gennaio)
mov bl,1
no_round:
mov bh,0           ; azzero bh perchè mi serve bx completo
mov al,[di+bx]     ; giorni del mese considerato
mov ah,0           ; azzero ah perchè mi serve ax completo per sommare la word vita
add vita,ax        ; aggiunge giorni dei mesi per ultimo anno
inc bl             ; passa al mese successivo
cmp bl,mese_m      ; mese_nascita != mese_morte ?
jne calcola_giorni_2 ; si --> cicla

jmp calcola_max    ; no --> calcolo terminato, vai al calcolo del massimo

continua_2:        ; mese_morte = mese_nascita
mov ax,anno_m
sub ax,anno_n      ; calcolo differenza anni morte-nascita
mov bx,365         ; giorni in un anno
xor dx,dx          ; azzero dx per la moltiplicazione
mul bx             ; (anno_m - anno_n)*365 --> giorni per gli anni vissuti
mov vita,ax        ; inizio a contare i giorni
mov al,giorno_m
sub al,giorno_n    ; giorno_morte - giorno_nascita
mov ah,0           ; azzero ah perchè mi serve ax completo per sommare la word vita
add vita,ax        ; aggiungi giorni

calcola_max:
mov ax,vita
cmp ax,vita_max    ; vita > vita_max ?
jbe nuova_persona  ; no --> nuova_persona
mov vita_max,ax    ; si --> nuovo massimo
mov dx,n           ; numero totale di persone
inc dx             ; incremento perchè cx nel loop è di una unità indietro
sub dx,cx          ; n_max = (n+1)-cx
mov n_max,dx       ; salvo il valore

nuova_persona:
add si,8           ; allineo data nascita della persona successiva
loop persona

mov ax,n_max       ; valore di ritorno

pop di
pop si
pop dx
pop cx
pop bx
pop bp

ret

_vecchio endp

calcolo_data proc  ;calcola da stringa di 8 caratteri i valori
                    ; numerici di giorno, mese e anno

push bp
mov bp,sp
push ax
push bx
push cx
push dx
push si
push di

xor bx,bx          ; azzero contatore
mov si,[bp+4]      ; si
mov al,[si]        ; primo carattere giorno
sub al,'0'         ; ascii --> binario
mov cl,10          ; devo moltiplicare per 10 così ottengo la
                    ; prima cifra decimale del giorno

mul cl
mov ah,[si+1]      ; secondo carattere giorno
sub ah,'0'         ; ascii --> binario
add ah,al          ; sommo le unità del giorno
mov bl,ah
mov [bp+6],bx      ; salvo il valore del giorno nello spazio riservatogli
mov al,[si+2]      ; primo carattere mese

```

```

sub al,'0'          ; ascii --> binario
mul cl              ; moltiplica per 10
mov ah,[si+3]       ; secondo carattere mese
sub ah,'0'          ; ascii --> binario
add ah,al           ; somma le unità
mov bl,ah
mov [bp+8],bx       ; salvo il valore del mese nello spazio riservatogli
mov al,[si+4]       ; primo carattere anno
sub al,'0'          ; ascii --> binario
mov ah,0            ; azzero ah per moltiplicazione
mov cx,1000         ; moltiplico per 1000
mul cx
mov bx,ax           ; bx conterrà l'anno tramite somme parziali di migliaia,
                   ; centinaia, decine e unità
mov al,[si+5]       ; secondo carattere anno
sub al,'0'          ; ascii --> binario
mov cl,100          ; moltiplico per 100
mul cl
add bx,ax           ; sommo alle migliaia
mov al,[si+6]       ; terzo carattere anno
sub al,'0'          ; ascii --> binario
mov ah,0
mov cl,10           ; moltiplico per 10
mul cl
add bx,ax           ; sommo decine
mov al,[si+7]       ; quarto e ultimo carattere anno
sub al,'0'          ; ascii --> binario
mov ah,0
add bx,ax           ; sommo unità
mov [bp+10],bx      ; salvo il valore dell'anno nello spazio riservatogli

pop di
pop si
pop dx
pop cx
pop bx
pop ax
pop bp

ret

calcolo_data endp

.exit
end

```

#### Soluzione Tema d'Esame 4

##### tde4.ASM

```

public _media_matrice

.model small
.stack
.data

somma dd 0

.code

_media_matrice proc
    push bp
    mov bp,sp
    push si
    push di
    push bx
    push cx
    push dx

    ;[bp+4]      *matrice
    ;[bp+6]      n
    ;[bp+8]      m
    ;[bp+10]     i
    ;[bp+12]     j

    mov bx,[bp+10]    ;bx = i
    dec bx            ;decremento riga perche' la
                    ;prima riga e' la riga 0
    mov ax,[bp+8]     ;ax = n_colonne
    cwd              ;preparo ax e dx alla moltiplicazione
    mul bx            ;ax = n_colonne*riga
    shl ax,1          ;ax = n_colonne*riga*2
                    ;--> punta al primo elemento della riga i-esima

    mov cx,[bp+8]     ;inizializzo contatore per scorrere la riga
    mov bx,ax         ;offset per scorrere la riga
    xor si,si         ;inizializzo a 0 si per scorrere la riga
    xor ax,ax         ;inizializzo a 0 ax per sommare i vari elementi
    add bx,[bp+4]     ;bx = indirizzo riga
row:
    mov ax,[bx][si]   ;sommo elemento j-esimo
    mov dx,word ptr somma ;parte meno significativa di somma
    add ax,dx         ;sommo ax
    mov word ptr somma,ax ;salvo la somma
    mov dx,word ptr somma+2 ;parte piu' significativa
    mov ax,0          ;sommo con 0
    adc ax,dx         ;sommo con carry
    mov word ptr somma+2,ax ;salvo la somma
    add si,2          ;passo a elemento successivo della riga
    loop row

    mov cx,[bp+6]     ;inizializzo contatore per scorrere la riga
    mov si,[bp+12]    ;inizializzo si al primo valore per scorrere la colonna
    dec si            ;prima colonna = colonna 0
    shl si,1          ;moltiplico per 2 perche' elementi di 16bit
    xor ax,ax         ;inizializzo a 0 ax per sommare i vari elementi
    mov bx,[bp+4]     ;indirizzo prima riga
col:
    add ax,[bx][si]   ;sommo elemento i-esimo
    mov dx,word ptr somma ;parte meno significativa di somma
    add ax,dx         ;sommo ax
    mov word ptr somma,ax ;salvo la somma
    mov dx,word ptr somma+2 ;parte piu' significativa
    mov ax,0          ;sommo con 0
    adc ax,dx         ;sommo con carry
    mov word ptr somma+2,ax ;salvo la somma
    mov dx,[bp+8]     ;numero elementi riga
    shl dx,1          ;moltiplico per 2 perche' elementi di 16bit
    add si,dx         ;passo al valore della riga successiva
    loop col

    mov bx,[bp+10]    ;bx = i
    dec bx            ;decremento riga perche' la
                    ;prima riga e' la riga 0
    mov ax,[bp+8]     ;ax = n_colonne
    cwd              ;preparo ax e dx alla moltiplicazione
    mul bx            ;ax = n_colonne*riga
    shl ax,1          ;ax = n_colonne*riga*2
                    ;--> punta al primo elemento della riga i-esima
    ;devo usare bx come base register
    ;bx = indirizzo riga elemento
    ;j
    dec si            ;prima colonna = colonna 0
    shl si,1          ;moltiplico per 2 perche' elementi di 16bit
    mov ax,[bx][si]   ;elemento i-j
    mov dx,word ptr somma ;parte meno significativa
    sub dx,ax         ;sottraggo elemento i-j alla somma degli altri elementi
    mov word ptr somma,dx ;salvo differenza
    mov ax,word ptr somma ;preparo ax e dx per la divisione
    mov dx,word ptr somma+2

```

##### tde4.C

```

#include <stdio.h>
extern int media_matrice (int *matrice, int n, int m, int i, int j);
void main()
{
    int matrice[20] = { 1, 2, 3, 4, 5,
                        6, 7, 8, 9,10,
                        11,12,13,14,15,
                        16,17,18,19,20};

    int n = 4;
    int m = 5;
    int i = 2;
    int j = 3;
    int k = 0;
    int q = 0;
    int media = 0;

    printf("Matrice: ");
    for (k=0; k<4; k++)
        for (q=0; q<5 ; q++)
        {
            printf("%d",matrice[k*m+q]);
            if(q==4)
                printf("\n");
        }
    media = media_matrice(matrice,n,m,i,j);

    printf("\nMedia: %d",media);
}

```

```

mov bx,[bp+8]      ;m
add bx,[bp+6]      ;n
dec bx             ;(m+n-1)
idiv bx            ;media in ax

pop dx
pop cx
pop bx
pop di
pop si
pop bp

ret

_media_matrice endp

.exit
end

```

Soluzione Tema d'Esame 5

tde5.ASM

```

public _qsort

.model small
.stack
.data

last dw 0

.code

_qsort proc near

;void qsort (int *v, int left, int right)

    push bp
    mov bp,sp
    push cx
    push bx
    push si
    push di
    mov di, [bp+6]
    mov si, [bp+4]

    cmp di, [bp+8]
    jge fine

    ;swap( v, left, (left+right)/2)
    mov ax, [bp+8]
    add ax, di
    cwd
    mov bx,2
    idiv bx

    push ax
    push di
    push si
    call _swap
    add sp,6

    mov last,di

    ;for( i=left+1; i<=right; i++) if( v[i]<v[left]) swap( v, ++last, i);
    mov bx,di
    inc bx
    cmp bx,[bp+8]
    jg fuori_for

    ciclo:
    push bx
    shl bx,1
    mov ax,[si+bx]
    mov bx,di
    shl bx,1
    cmp ax,[si+bx]
    jge continua

    ;push bx
    inc last
    push last
    push si
    call _swap
    add sp,4

    continua:
    pop bx
    inc bx
    cmp bx,[bp+8]
    jle ciclo

    fuori_for:
    ;swap( v, left, last)
    push last
    push di
    push si
    call _swap
    add sp,6

    ;qsort( v, left, last)
    push last
    push di
    push si
    call _qsort
    add sp,6

    ;qsort( v, last+1, right);
    push [bp+8]
    mov cx,last
    inc cx
    push cx
    push si
    call _qsort
    add sp,6

    fine:
    pop di
    pop si
    pop bx
    pop cx
    pop bp
    ret

_qsort endp

_swap proc

    push bp
    mov bp, sp
    push dx
    push bx
    push si
    push di

    mov bx, [bp+4]
    mov si, [bp+6]
    mov di, [bp+8]
    shl si, 1
    shl di, 1

    ;*v
    ;primo elemento
    ;secondo elemento
    ;moltiplico per 2 visto che word da 16bit

```

tde5.C

```

#include <stdio.h>
extern void qsort (int *v,int left, int right);
void main()
{
    int v[] = {12,56,34,42,20,26};
    int left = 0;
    int right = 5;

    printf("Vettore iniziale: ");
    for (int i=0; i<6; i++)
        printf("%d",v[i]);

    qsort(v,left,right);

    printf("\nVettore ordinato: ");
    for (int i=0; i<6; i++)
        printf("%d",v[i]);
}

```

```

        mov dx, [bx][si]          ;dx valore d'appoggio
        xchg dx, [bx][di]        ;scambio
        mov [bx][si], dx

        pop di
        pop si
        pop bx
        pop dx
        pop bp
        ret

_swap endp

.exit
end

```

## Soluzione Tema d'Esame 6

### tde6.ASM

```

public _substitute

.model small
.stack
.data

sostituzioni dw 0
start_sost dw 0
str1_len dw 0
rew_len dw 0

.code

_substitute proc
    push bp
    mov bp,sp
    push si
    push di
    push bx
    push cx
    push dx

    ;[bp+4]          ;*text
    ;[bp+6]          ;*str1
    ;[bp+8]          ;*str2

    mov bx,[bp+6]    ;str1
    xor si,si
    length:
    cmp [bx+si],0    ;fine stringa?
    je inizio        ;si --> inizio
    inc si
    inc str1_len      ;no --> incrementa str1_len
    jmp length

    inizio:
    mov bx,[bp+4]    ;*text
    mov dx,[bp+8]    ;*str2

    xor di,di        ;inizializzazione offset per spostamento sulla stringa text

    scorri_testo:
    xor si,si        ;inizializzazione offset per spostamento sulla stringa str1
    mov cx,[bp+6]    ;*str1
    mov al,[bx+di]   ;al = primo carattere di text
    cmp al,0         ;carattere di fine stringa di text?
    je fine          ;si --> fine
    push bx          ;salvo bx nello stack
    mov bx,cx        ;bx = *str1
    mov ah,[bx+si]   ;ah = primo carattere di str1
    pop bx           ;ricarico valore di bx
    cmp al,ah        ;caratteri uguali?
    jz scorri_stringa ;si --> scorri_stringa
    inc di           ;no --> passa al carattere successivo di text
    jmp scorri_testo

    scorri_stringa:
    mov start_sost,di ;possibile offset di sostituzione

    ciclo:
    inc di           ;carattere successivo di text
    inc si           ;carattere successivo di str1
    push bx          ;salvo bx nello stack
    mov bx,cx        ;bx = *str1
    mov ah,[bx+si]   ;ah = primo carattere di str1
    pop bx           ;ricarico valore di bx
    cmp ah,0         ;carattere di fine stringa di str1?
    je sostituisci   ;si --> sostituisci
    mov al,[bx+di]   ;al = primo carattere di text
    cmp al,0         ;carattere di fine stringa di text?
    je fine          ;si --> fine
    cmp al,ah        ;caratteri uguali?
    jz ciclo         ;si --> ciclo
    inc di           ;no --> passa al carattere successivo
    jmp scorri_testo ;e vai a scorri_testo

    sostituisci:
    xor si,si
    xor ax,ax

    mov di,start_sost ;offset str1 in text
    add di,str1_len
    mov rew_len,0
    salva:
    inc rew_len
    mov al,[bx+di]   ;salvo carattere nello stack
    push ax
    inc di
    cmp al,0         ;carattere di fine stringa?
    jne salva        ;no --> continua a salvare

    add start_sost,bx ;start_sost = indirizzo possibile sostituzione
    mov di,start_sost ;di = indirizzo primo carattere da sostituire

    ciclo_sost:
    push bx          ;salvo bx nello stack
    mov bx,dx        ;bx = *str2
    mov al,[bx+si]   ;al = primo carattere di str2
    pop bx           ;ricarico valore di bx
    cmp al,0         ;carattere di fine stringa di str2?
    je end_sost      ;si --> end_sost
    mov [di],al      ;sostituisco carattere
    inc si           ;passa al carattere successivo di str2
    inc di           ;passa al carattere successivo da sostituire
    jmp ciclo_sost

    end_sost:
    inc sostituzioni ;incrementa contatore sostituzioni
    sub di,bx        ;di offset da cui ripartire la ricerca
    ;inc di          ;passa al carattere successivo di text

    mov si,di
    add si,rew_len   ;parto dalla fine per rimettere i caratteri
    dec si
    mov cx,rew_len   ;caratteri da scrivere
    scrivi:
    pop ax
    mov [bx+si],al
    dec si
    loop scrivi

```

### tde6.C

```

#include <stdio.h>

extern int substitute (char *text, char *str1, char *str2);

int main(void)
{
    char text[30] = "xxxciaocc";
    char str1[30] = "ciao";
    char str2[30] = "hola";
    int sostituzion;
    printf("text: %s",text);
    printf("\nstr1: %s",str1);
    printf("\nstr2: %s",str1);
    sostituzion=substitute(text, str1, str2);
    printf("\n\ntext dopo sostituzione: %s",text);
    printf("\nNumero di sostituzioni: %d",sostituzion);
    return(0);
}

```

```
    jmp scorri_testo

fine:
mov ax,sostituzioni ;valore di ritorno

pop dx
pop cx
pop bx
pop di
pop si
pop bp

ret

_substitute endp

.exit
end
```

[Edit](#) - [History](#) - [Print](#) - [Recent Changes](#) - [Search](#)

Page last modified on November 12, 2008, at 10:35 AM