Esami con soluzione

# Esame 1

Esercizio di assembly Scrivere in Assembler per Intel 80x86 la funzione ModificaStringa che riceve in ingresso una stringa zero-terminata Sorg (contenente solo caratteri alfabetici maiuscoli), una seconda stringa Dest e un vettore di byte sia positivi che negativi (con lo stesso numero di valori dei caratteri della stringa Sorg). I parametri devono essere passati mediante lo stack. La funzione deve copiare nella stringa Dest i caratteri della stringa Sorg diminuiti del valore corrispondente in Vett, verificando che il carattere risultante sia compreso tra ‘A’ e ‘Z’ (inclusi). Se il valore risultante è “inferiore” a ‘A’ deve copiare in Dest un carattere ‘-‘ (meno) e se il valore risultante è “superiore” a ‘Z’ deve copiare un carattere ‘+’ (più). Ad esempio, se le variabili del programma fossero le seguenti: Sorg: db ‘BUONESAME’,0 Dest: resb 100 Vett: db 3,-2,10,-2,0,-10,1,1,2 Il risultato in Dest dovrebbe essere: ‘-WEPE+-LC’ (es. B-3 è inferiore a A quindi metto -; S-(-10) è superiore a Z quindi metto +; ecc.) Si scriva anche il programma main che chiama la funzione e che alla fine deve scrivere sullo schermo la stringa Dest

SECTION data

Sorg: db 'BUONESAME',0

Dest: resb 100

Vett: db 3,-2,10,-2,0,-10,1,1,2

SECTION text

..start:

mov ax, data mov ds, ax mov es, ax mov ax, Sorg push ax

mov ax, Dest push ax

mov ax, Vett

push ax

call ModificaStringa add sp, 6

mov ax, Dest push ax

call ScriviStringa add sp, 2

mov ax, 4c00h

int 21h

ModificaStringa: push bp

mov bp, sp

mov si, [bp+8] ; Sorg mov di, [bp+6] ; Dest mov bx, [bp+4] ; Vett

Ciclo:

lodsb

cmp al, 0 je Fine

mov ah, [bx]

sub al, ah cmp al, 'A'

jl mettimeno ; con la differenza diventa "inferiore" a 'A'

cmp al, 'Z'

jg mettipiu ; con la differenza diventa "superiore" a 'Z' salta:

stosb ; compreso tra 'A' e 'Z' quindi memorizzo in Dest inc bx

jmp Ciclo mettimeno:

mov al, '-'

jmp salta mettipiu:

mov al, '+'

jmp salta Fine:

stosb pop bp ret

ScriviStringa: push bp mov bp, sp

mov si,[bp+4] mov ah,0eh mov bx,0000h

Stampa:

lodsb

cmp al, 0

je fineStampa int 10h

jmp Stampa fineStampa:

pop bp

ret

;Ritorno alla procedura chiamante

# Esame 2

Esercizio di assembly Scrivere in Assembler per Intel 80x86 la funzione CalcolaValori che riceve in ingresso tre vettori di word (16 bit), V1, V2 e V3, e un byte N positivo che indica la lunghezza dei vettori (uguale per tutti e tre). I parametri devono essere passati mediante lo stack. La funzione deve restituire nel registro AX il numero di volte che il valore V1[i]+V2[N-i-1] è maggiore del valore V3[i]. In pratica, si deve scorrere i vettori V1 e V3 da sinistra a destra, mentre il vettore V2 da destra a sinistra partendo dal fondo. Ad esempio, se le variabili del programma fossero le seguenti: V1: dw 3, 7, -21, 22, 6 V2: dw 9, 8, 22, 7, -9 V3: dw -7, 15, 0, 31, 12 N: db 5 Il risultato in AX sarebbe 3. Infatti: V1[0]+V2[4]=3+(-9)=-6 è maggiore di V3[0]=-7 --> +1 V1[1]+V2[3]=7+7=14 è minore di V3[1]=15

V1[2]+V2[2]=-21+22=1 è maggiore di V3[2]=0 --> +1 V1[3]+V2[1]=22+8=30 è minore di V3[3]=31 V1[4]+V2[0]=6+9=15 è maggiore di V3[4]=12 --> +1 Si scriva

anche il programma main che chiama la funzione. Si verifichi con il debug il valore finale di AX (dopo la chiamata) per verificare la correttezza della soluzione

SECTION data

V1: dw 3, 7, -21, 22, 6

V2: dw 9, 8, 22, 7, -9

V3: dw -7, 15, 0, 31, 12

N: db 5

SECTION text

..start:

mov ax, data mov ds, ax mov es, ax mov ax, V1 push ax

mov ax, V2 push ax mov ax, V3 push ax xor ah, ah

mov al, [N] push ax

call CalcolaValori add sp, 8

mov ax, 4c00h int 21h

CalcolaValori: push bp mov bp, sp

mov si, [bp+10] ; V1 mov di, [bp+8] ; V2

add di, [bp+4] ; sposto di su V2[N]

add di, [bp+4] ; sommo due volte N perchè sono word!! Quindi ogni valore occupa due byte dec di

dec di ; decremento di 2 per puntare a V2[N-1] mov bx, [bp+6] ; V3

mov cx, [bp+4] ; N xor dx, dx

Ciclo:

lodsw

add ax, [di] cmp ax, [bx] jle NonMaggiore

inc dx ; uso dx temporaneo per contare NonMaggiore:

inc bx

inc bx ; sposto avanti bx per puntare alla successiva word di V3 dec di

dec di ; sposto indietro di per puntare alla precedente word di V2

; nota: per si non devo farlo perchè lo fa già lodsw loop Ciclo

mov ax, dx

pop bp ret

# Esame 3

Esercizio di assembly Scrivere in Assembler per Intel 80x86 la funzione ContaSeSuperioreMedia che riceve in ingresso due vettori di byte positivi, V1 e V2 (di lunghezza N fornito come dato), e un byte Nmax positivo. La funzione deve verificare se i valori di V2 superiori alla media dei valori di V1 (media arrotondata per

difetto) sono superiori o uguali a Nmax (e in questo caso scrivere a video la stringa S1) o no (e in questo caso scrivere a video la stringa S2). Tutti i 6 parametri devono essere passati mediante lo stack. Le stringhe S1 e S2 sono zero-terminate secondo la prassi C. Ad esempio, se le variabili del programma fossero le seguenti: N: db 5 V1: db 3, 7, 21, 22, 6 V2: db 9, 2, 22, 11, 19 Nmax: db 4 S1: db "Valori superiori alla media maggiori di o uguali a Nmax",0 S2: db "Valori superioti alla media minori di Nmax",0 Verrebbe scritta la stringa S1 in quanto la media dei valori di V1 è 3 (somma=59 che diviso per 5 restituisce 11 con resto di 4) e il numero di valori in V2 superiori o uguali a 11 sono 2 (22 e 19). Visto che Nmax vale 4 si deve scrivere la stringa S2. Si scriva anche il programma main che chiama la funzione

SECTION data N: db 5

V1: db 3, 7, 21, 22, 6

V2: db 9, 2, 22, 11, 19

Nmax: db 4

S1: db "Valori superiori alla media maggiori di o uguali a Nmax",0 S2: db "Valori superioti alla media minori di Nmax",0

SECTION text

..start:

mov ax, data mov ds, ax mov es, ax xor ah, ah mov al, [N] push ax

mov al, [Nmax] push ax

mov ax, V1

push ax mov ax, V2 push ax mov ax, S1 push ax mov ax, S2 push ax

call ContaSeSuperioreMedia add sp, 12

mov ax, 4c00h int 21h

ContaSeSuperioreMedia: push bp

mov bp, sp

mov si, [bp+10] ; V1 mov di, [bp+8] ; V2 xor ch, ch

mov cl, [bp+14] ; N

xor dx, dx xor ah, ah

Ciclo:

lodsb

add dx, ax loop Ciclo

mov cl, [bp+14] ; N mov ax, dx

xor dx, dx div cx

xor dl, dl Ciclo2:

cmp [di], al

jle Inferiore inc dl

Inferiore:

inc di

loop Ciclo2

cmp dl, [bp+12] jge ScriviS1 mov ax, [bp+4] jmp Fine

ScriviS1:

mov ax, [bp+6] Fine:

push ax

call ScriviStringa add sp, 2

pop bp

ret

ScriviStringa: push bp mov bp, sp

mov si,[bp+4]

mov ah,0eh mov bx,0000h

Stampa:

lodsb

cmp al, 0

je fineStampa int 10h

jmp Stampa fineStampa:

pop bp

ret

;Ritorno alla procedura chiamante

# Esame 4

Esercizio di assembly Scrivere in Assembler per Intel 80x86 un programma che fa inserire all’utente da tastiera una serie di caratteri che termina quando l’utente preme il pulsante ESC. Il programma, tramite una funzione LeggiEModifica riceve, uno ad uno, i caratteri inseriti dall’utente e li modifica aumentando il loro codice ASCII di un valore che aumenta ad ogni carattere (partendo da zero). La nuova stringa così modificata va inserita nella stringa zeroterminata (secondo la convenzione C) Stringa. In pratica, se l’utente inserisse i caratteri: Ben tornati dalle ferie! nella Stringa verrebbe inserito Bfp#xtxui}s+pnz{u1xxå~{8 (B+0=B e+1=f n+2=p ...) Il programma deve poi visualizzare a schermo la stringa risultante e memorizzare in AX il numero di vocali (considerando solo le standard a,e,i,u,o) nella Stringa modificata (nell’esempio in AX va il valore 3 – u, i, u). I parametri devono essere passati alla funzione mediante lo stack. Si scriva anche il programma main che chiama la funzione.

SECTION data Stringa: resb 100

SECTION text

..start:

mov ax, data mov ds, ax

xor cl, cl ; conta i caratteri letti xor ch, ch ; conta le vocali

CicloLettura:

mov ah, 00h int 16h

cmp al, 1bh

je Fine push ax

mov ax, Stringa push ax

call LeggiEModifica

add sp, 4 inc cl

jmp CicloLettura

Fine:

; ZERO TERMINA STRINGA

xor ah, ah mov al, cl add si, ax

mov [si+1], byte 0

; SCRIVI STRINGA

mov ax, Stringa push ax

call ScriviStringa

add sp, 2

mov ax, 4c00h int 21h

LeggiEModifica: push bp

mov bp, sp

mov si, [bp+4] ; Stringa

mov dl, [bp+6] ; carattere letto xor ah, ah

mov al, cl

add dl, cl add si, ax mov [si], dl

call ControllaVocale pop bp

ret

ControllaVocale: cmp dl, 'a' je ok

cmp dl, 'e'

je ok

cmp dl, 'i'

SECTION data

Stringa: db 'ArchItEtTo'

SECTION text

..start:

mov ax, data mov ds, ax

xor cl, cl ; per ricordare se maiuscolo=0 o minuscolo=FF

CicloLettura:

mov ah, 00h int 16h

cmp al, 1bh je Fine

xor ah,ah

push ax

mov ax, Stringa push ax

call VisualizzaCaratteri add sp, 4

jmp CicloLettura Fine:

mov ax, 4c00h

int 21h

VisualizzaCaratteri: push bp

mov bp, sp

mov si, [bp+4] ; Stringa

mov bx, [bp+6] ; carattere letto

sub bx, 30h ; ottengo il numero dal codice ASCII mov al, [si+bx] ; carattere da stampare

cmp al, 61h ; se maggiore è una lettera minuscola

je ok

cmp dl, 'o' je ok

cmp dl, 'u' je ok

cmp dl, 'A'

je ok

cmp dl, 'E' je ok

cmp dl, 'I' je ok

cmp dl, 'O' je ok

cmp dl, 'O'

je ok ret

ok:

inc ch ret

ScriviStringa: push bp mov bp, sp

mov si,[bp+4] mov ah,0eh mov bx,0000h

Stampa:

lodsb

cmp al, 0

je fineStampa int 10h

jmp Stampa

fineStampa:

pop bp ret

;Ritorno alla procedura chiamante

# Esame 5

Esercizio di assembly Scrivere in Assembler per Intel 80x86 un programma che fa inserire all’utente da tastiera un numero da 0 a 9 (singola cifra), iterativamente fintanto che l’utente non preme il pulsante ESC. Il programma, tramite una funzione VisualizzaCaratteri riceve, uno ad uno, i numeri inseriti dall’utente e stampa a video il carattere della stringa Stringa corrispondente alla posizione inserita (si supponga la stringa di almeno 10 caratteri e contenente solo lettere maiuscole o minuscole). Il carattere così selezionato deve però essere visualizzato alternativamente in maiuscolo e minuscolo (partendo da maiuscolo). Ad esempio, se la stringa valesse: ArchItEtTo e l’utente premesse in sequenza 0 3 2 4 9 1 7 0, sullo schermo verrebbero visualizzati i seguenti caratteri: AhCiOrTa (pos. 0 - A maiuscola; pos. 3 – h minuscola; pos. 2 – c maiuscola, ...) I parametri devono essere passati alla funzione mediante lo stack. Si scriva anche il programma main che chiama la funzione

jae Minuscola

Maiuscola:

cmp cl, 0 je ok

add al, 20h jmp ok

Minuscola:

cmp cl, 0 jne ok

sub al, 20h

ok:

mov ah,0eh mov bx,0000h int 10h

not cl pop bp ret

; devo visualizzare maiuscola ed è maiuscola

; devo visualizzare minuscola ma è maiuscola --> +20h

; devo visualizzare minuscolo ed è minuscola

; devo visualizzare maiuscolo ma è minuscola --> -20h

# Esame 6

Esercizio di assembly Scrivere in Assembler per Intel 80x86 un programma che fa inserire all’utente da tastiera un numero da 0 a 9 (singola cifra), iterativamente fintanto che l’utente non preme il pulsante ESC. Il programma, tramite la funzione CopiaAlternata riceve, uno ad uno, i numeri inseriti (il numero viene passato mediante lo stack) e esegue le seguenti operazioni: ◻ Controlla se il numero è pari o dispari ◻ Se il numero è pari copia in una stringa destinazione Dest dichiarati tra i dati del programma il prossimo carattere della stringa sorgente Sorg1 ◻ Se il numero è dispari copia invece il prossimo carattere della stringa sorgente Sorg2 Tutte le stringhe sono zero-terminate secondo al convenzione C. Se la stringa sorgente scelta dovesse essere terminata (non avere più caratteri a disposizione) il programma procede con l’altra stringa. Se entrambe le stringhe fossero terminate e l’utente continuasse a premere un carattere diverso da ESC, il programma deve uscire. Ad esempio, se le variabili del programma fossero le seguenti: Sorg1: db “Pera”,0 Sorg2: db “Ascia”,0 Dest: resb 100 e l’utente premesse in sequenza 2 3 5 2 0 7 6 2 3 2, nella variabile Dest dovrebbe essere inserito (zero-terminato!) il seguente valore: PAsercaia Infatti: 2 (pari) ◻ copia ‘P’ da Sorg1 3 (dispari) ◻ copia ‘A’ da Sorg2 5 (dispari) ◻ copia ‘s’ da Sorg2 2 (pari) ◻ copia ‘e’ da Sorg1 0 (pari) ◻ copia ‘r’ da Sorg1 7 (dispari) ◻ copia ‘c’ da Sorg2 6 (pari) ◻ copia ‘a’ da Sorg1 2 (pari) ◻ la stringa Sorg1 è finite quindi copia ‘i’ da Sorg2 3 (dispari) ◻ copia ‘a’ da Sorg2 2 (pari) ◻ entrambe le stringhe sono finite quindi il programma termina Si scriva anche il programma main che chiama la funzione.

SECTION data Sorg1: db 'Pera',0

Sorg2: db 'Ascia',0

Dest: resb 100

SECTION text

..start:

mov ax, data mov ds, ax mov es, ax mov si, Sorg1 mov bx, Sorg2 mov di, Dest

CicloLettura:

mov ah, 00h int 16h

cmp al, 1bh

je Fine xor ah,ah push ax

call CopiaAlternata add sp, 2

jmp CicloLettura Fine:

mov [di], byte 0

mov di, Dest push di

call ScriviStringa add sp, 2

mov ax, 4c00h

int 21h

CopiaAlternata: push bp

mov bp, sp

mov ax, [bp+4] ; carattere letto

sub ax, 30h ; ottengo il numero dal codice ASCII test al, 00000001b; verifico se pari o dispari

je Pari

Dispari: ; devo prendere da Sorg2 mov cl, [bx]

cmp cl, 0 ; verifico se ho ancora elementi in Sorg2 je FineSorg2

inc bx

jmp FineFunzione

Pari: ; devo prendere da Sorg1

SECTION data

Vett: db 3, -5, 7, 12, -3, 11, 22

N: db 7

SECTION text

..start:

mov ax, data mov ds, ax mov ax, Vett push ax

xor ah, ah mov al, [N] push ax

call ScambiaValori add sp, 4

mov ax, 4c00h int 21h

ScambiaValori: push bp mov bp, sp

mov cx, [bp+4] ; N

dec cx ; NOTA: l'ultimo valore non va scambiato in ogni caso mov si, [bp+6] ; Vett

Ciclo:

mov ah, 00h int 16h

mov cl, [si]

cmp cl, 0 je Dispari inc si

jmp FineFunzione FineSorg2:

mov cl, [si] cmp cl, 0

je Fine inc si

jmp FineFunzione

FineSorg1:

mov cl, [bx] cmp cl, 0

je Fine inc bx

FineFunzione:

mov [di], cl inc di

pop bp

ret

; verifico se ho ancora elementi in Sorg1

; se è finito Sorg1, copio Sorg2

; se è finito Sorg2, copio Sorg1

; verifico se ho ancora elementi in Sorg1

; sono finite entrambe, esci! USCITA UN PO' BRUSCA, MA OK

; se è finito Sorg1, copio Sorg2

; verifico se ho ancora elementi in Sorg2

; sono finite entrambe, esci! USCITA UN PO' BRUSCA, MA OK

ScriviStringa:

push bp mov bp, sp

mov si,[bp+4] mov ah,0eh mov bx,0000h

Stampa:

lodsb

cmp al, 0

je fineStampa int 10h

jmp Stampa fineStampa:

pop bp

ret

;Ritorno alla procedura chiamante

# Esame 7

Esercizio di assembly Scrivere in Assembler per Intel 80x86 un programma che scorre un vettore di N byte Vett e, ad ogni passo, richiede all’utente di inserire un carattere tra ‘S’ (codice ASCII 83) e ‘N’ (codice ASCII 78). Se l’utente preme ‘S’ l’elemento attualmente puntato su Vett deve essere scambiato con il successive, altrimenti no. La richiesta all’utente continua iterativamente fintanto o l’utente preme il pulsante ESC o si arriva alla fine del vettore Vett. Se l’utente preme ‘N’ i valori di Vett non vanno scambiati ma si passa al prossimo elemento di Vett, mentre se l’utente preme un qualsiasi altro tasto il programma deve chiedere nuovamente il tasto all’utente rimanendo nell’elemento corrente di Vett. Il programma principale deve chiamare una funzione ScambiaValori a cui passa mediante lo stack l’indirizzo di Vett e il valore di N. Ad esempio, se le variabili del programma fossero le seguenti: Vett: db 3, -5, 7, 12, -3, 11, 22 N: db 7 e l’utente premesse in sequenza SSNSNS, nella variabile Vett al termine del programma ci sarebbero i seguenti valori: -5, 7, 3, -3, 12, 22, 11 Si scriva anche il programma main che chiama la funzione.

cmp al, 1bh

je Fine cmp al, 83 je Scambio cmp al, 78 jne Ciclo inc si loop Ciclo jmp Fine

Scambio:

lodsb

mov ah, [si] mov [si], al mov [si-1], ah loop Ciclo

Fine:

pop bp ret

; l'utente ha premuto S

; l'utente ha premuto N

; mi sposto al prossimo carattere

; decrementa anche CX cioè N

; elemento corrente in AL

; prossimo elemento in AH - NOTA: lodsb ha già incrementanto SI

; decrementa anche CX cioè N