UNIVERSITÀ
di VERONA

Università degli Studi di Verona
Facoltà di Scienze e Ingegneria
AA 2018/2019
Corso di laurea in Informatica

Elaborato Assembly

Laboratorio di Architettura degli Elaboratori

Besoli Samuele - Caldana Stefano
VR437603 - VR437143

Introduzione

Il progetto è formato da due file sorgenti:

- Un file principale denominato `GestioneVettore.s`, all'interno del quale viene effettuata l'operazione di inserimento dei valori del vettore, la stampa del menù contenente le opzioni disponibili e la gestione delle richieste da parte dell'utente.
- Un file denominato `Array.s` contenente l'implementazione delle funzioni, che eseguono sul vettore le operazioni disponibili nel menù.

Scelte progettuali

- Al contrario di come sarebbe preferibile fare in linguaggio C, in Assembly l'algoritmo si dimostra più prestante senza la funzione `scegliOpzioni` e utilizzando solamente etichette. Questo il motivo per cui vengono utilizzate esclusivamente le etichette per la scelta dell'operazione.
- Il corpo del programma si trova all'interno del file `GestioneVettore.s` mentre le funzioni sono state separate in un file esterno chiamato `Array.s` in modo da poter utilizzare queste funzioni come una specie di libreria.
- Lettura e scrittura del vettore vengono effettuate tramite puntatore per una migliore performance.
- Le funzioni contenute in `Array.s` ricevono la dimensione del vettore. E' posibile quindi riutilizzare a proprio piacimento le varie funzioni.
- Lettura del vettore con uno shift al valore successivo con il comando `pushl (%ebx, %esi, 4)` dove in `%ebx` è presente il puntatore al primo valore, in `%esi` l'offset ovvero di quanto è lo shift (%esi*4) e nell'ultima "casella" un valore che sia 2^n per la moltiplicazione con il secondo elemento. Tutte queste operazioni si trovano all'interno della parentesi perché prende in memoria il valore contenuto nell'indirizzo.

```
# 2 - stampa a video del vettore inserito in ordine inverso
stampa_inv_ext:
    movl %ecx, %esi
    ciclo_stampa_inv:
        decl %esi
        pushl (%ebx, %esi, 4)       # dimostrazione utilizzo del comando
        pushl $char_stampa
        call printf
        addl $8, %esp
        cmp $0, %esi
    jg ciclo_stampa_inv
    call flush
ret
```

- Implementazione di una funzione chiamata `flush` che serve a pulire il buffer e permettere così la visione nel terminale quando viene utilizzata la funzione `printf()`.

Variabili

GestioneVettore.s

Nome	Tipo	Descrizione
vet	.long[]	E' un vettore di n elementi, inizializzato a 0, il quale verrà poi riempito con i valori inseriti dall'utente, e sul quale verranno eseguite le operazioni richieste.
length	.long	Dimensione del vettore
scelta	.long	Variabile utilizzata per la scelta dell'operazione da eseguire
scanf_int	.string	Variabile di tipo stringa contente il tipo di valore da acquisire
input_value	.long	Variabile utilizzata nelle funzioni che richiedono un valore in input
*_message	.string	Variabili di tipo stringa, utilizzate per la stampa di messaggi relativi alle funzioni contenute nel programma

Array.s

Nome	Tipo	Descrizione
title line opz1 .. opz10	.string	Variabili di tipo stringa, utilizzate per stampare i tipi di funzioni disponibili nel menu principale del programma
char_stampa	.string	Variabile di tipo stringa contenente il tipo e il formato del valore da stampare
scanf_int	.string	Variabile di tipo stringa contente il tipo del valore da acquisire
fattore	.long	Variabile con valore di $4 (long) necessaria per l'azzeramento dell'array nella ricerca del valore più frequente
num_pari num_dispari	.string	Variabili di tipo stringa utilizzate nella funzione di conteggio degli elementi pari/dispari
input_message	.string	Variabile di tipo stringa utilizzata per stampare la richiesta di inserimento del valore
vet_dim	.long	Variabile utilizzata per salvare la lunghezza del vettore

Modalità di passaggio e restituzione dei valori delle funzioni create

Tutte le funzioni sono riutilizzabili in quanto ad esse viene "passato" il puntatore al vettore e la sua dimensione, quindi il programma funziona con qualsiasi dimensione del vettore.

Per il passaggio dei parametri alle funzioni, sono stati utilizzati i registri, in particolare il registro `%ebx` è stato utilizzato per il puntatore al vettore, mentre il registro `%ecx` per la lunghezza del vettore stesso.

```
# 1 - stampa a video del vettore inserito
stampa_ord:
    leal vet, %ebx
    movl length, %ecx
    call stampa_ord_ext
ret
```

Nella restituzione dei valori, tutte le funzioni che necessitano di restituire un valore, lo restituiscono tramite il registro `%eax`.

```
# 5 - stampa il massimo valore inserito
trova_max_ext:
    movl (%ebx), %eax
    addl $4, %ebx
    decl %ecx
    ciclo_trova_max:
        cmp %eax, (%ebx)
        jle continua_trova_max
            movl (%ebx), %eax   # %eax contiene il valore massimo inserito
        continua_trova_max:
        addl $4, %ebx
    loop ciclo_trova_max
ret
```

Pseudo-codice

GestioneArray.s

```
function _start()
    length = int 10
    scelta = int 0
    input_value = int 0
    vet = int[length]

    inserimento_ext(vet, length)
    stampa_opz()

    do{
        printf(input_message)
        scanf(&scelta)
        switch(scelta)
            case 0:
                printf(exit_message)
                break
            case 1:
                stampa_opz_ext(&vet, length)
                break
            case 2:
                stampa_inv_ext(&vet, length)
                break
            case 3:
                conta_pari_dispari(&vet, length)
                break
            case 4:
                printf(input_valore_message)
                scanf(&input_value)
                pos = trova_valore_ext(&vet,length,input_value)
                if(val == -1)
                    printf(not_found_message)
                else
                    printf(found_message, pos)
                break
            case 5:
                val = trova_max_ext(&vet,length)
                printf(trova_max_message,val)
                break
            case 6:
                pos = posizione_max_ext(&vet,length)
                printf(posizione_max_message,pos)
                break
            case 7:
                val = trova_min_ext(&vet,length)
                printf(trova_min_message,val)
                break
            case 8:
                pos = posizione_min_ext(&vet,length)
                printf(posizione_min_message,pos)
                break
```

```
            case 9:
                val = frequenza_max_ext(&vet,length)
                printf(valore_freq_message,val)
                break
            case 10:
                val = valore_medio_ext(&vet,length)
                printf(media_message)
                break
            default
                printf(error_message)
                break
    return
    } while(scelta != 0)

    exit(0)
```

Array.s

```
function inserimento_ext(*vet, length)
    input_message = "Inserire l'intero in posizione %d: "
    for(i=0; i<length; i++)
        printf(input_message)
        scanf(vet[i])
    return
```

```
function stampa_opz_ext()
    printf(title)
    printf(line)
    printf(opz1)
    printf(opz2)
    printf(opz3)
    printf(opz4)
    printf(opz5)
    printf(opz6)
    printf(opz7)
    printf(opz8)
    printf(opz9)
    printf(opz10)
    return
```

```
function stampa_ord_ext(*vet, length)
    char_stampa = "%3d"
    for(i = 0; i < length; i++)
        printf(char_stampa,vet[i])
    return
```

```
function stampa_inv_ext(*vet, length)
    char_stampa = "%3d"
    for(i = length; i >= 0; i--)
        printf(char_stampa,vet[i])
    return
```

```
function isPari(num)
    if ( num % 2 == 0 )
        return 1
    else
        return 0
```

```
function conta_pari_dispari_ext(*vet, length)
    num_pari = "Pari: %d \n"
    num_dispari = "Dispari: %d \n"
    dispari = 0
    pari = 0
    for (i=0; i < length; i++)
        if (isPari(vet[i]) == 1)
            pari += 1
        else
            dispari += 1
    printf(num_pari, pari)
    printf(num_dispari, dispari)
    return
```

```
function trova_valore_ext(*vet, length, input_value)
    for(i=0; i<length; i++)
        if ( input_value == vet[i] )
            return i
    return -1
```

```
function trova_max_ext(*vet, length)
    max = vet[0];
    for(i=1; i<length; i++)
        if(vet[i] > max)
            max = vet[i]
    return max
```

```
function posizione_max_ext(*vet, length)
    max = vet[0]
    pos = 0
    for(i=1; i<length; i++)
        if( vet[i] > max )
            max = vet[i]
            pos = i
    return pos
```

```
function trova_min_ext(*vet, length)
    min = vet[0]
    for(i=1; i<length; i++)
        if(vet[i] < min)
            min = vet[i]
    return min
```

```
function posizione_min_ext(*vet, length)
    min = vet[0]
    pos = 0
    for(i=1; i<length; i++)
        if( vet[i] < min )
            min = vet[i]
            pos = i
    return pos
```

```
function frequenza_max_ext(*vet, length)
    maxFreq = 0
    valFreq = 0
    freq = 0
    for(i=0; i<length; i++)
        for(j=0; j<length; j++)
            if(vet[i] == vet[j])
                freq += 1
        if(freq > maxFreq)
            maxFreq = freq
            valFreq = vet[i]
    return valFreq
```

```
function valore_medio_ext(*vet, length)
    sum = 0
    for(i=0; i<length; i++)
        sum += vet[i]
    return (int)(sum/length)
```