Esercizi Assembly 2

M. Sonza Reorda – M. Grosso

Politecnico di Torino
Dipartimento di Automatica e Informatica

Esercizio 1

- Dato un vettore di DIM word in memoria, rimpiazzarlo con il vettore inverso (senza usare un altro vettore di appoggio).

prima
423
3191
23
11
-412
3
9

dopo
9
3
-412
11
23
3191
423

Implementazione

- Non è possibile effettuare operazioni tra due indirizzi di memoria
- Utilizziamo:
  - il registro CX come contatore
  - il registro SI per l'indirizzo "basso"
  - il registro DI per l'indirizzo "alto"
  - il registro AX per il dato temporaneo.

Codice

```
DIM EQU 7

.model small
.stack
.data

vettore dw 423, 3191, 23, 11, -412, 3, 9

.code
.startup

mov cx, DIM/2        ; calcolato @compile-time; funziona con DIM pari o dispari
xor si, si           ; azzeramento indice elemento origine
mov di, (DIM-1)*2    ; calcolo indice dell'ultimo elemnento (@compile-time)
```

Codice [cont.]

```
ciclo: mov ax, vettore[si]
       xchg vettore[di], ax
       mov vettore[si], ax
       add si, 2
       sub di, 2
       dec cx
       jnz ciclo

.exit
end
```

Implementazione alternativa

- Utilizzo lo *stack*:
  - Prima leggo il vettore ed eseguo DIM operazioni di *push*
  - Dopo eseguo DIM operazioni di *pop* e scrivo il vettore.

Codice

```
DIM EQU 7

.model small
.stack
.data

vettore dw 423, 3191, 23, 11, -412, 3, 9

.code
.startup

mov cx, DIM
xor si, si
```

Codice [cont.]

```
ciclo1:  push vettore[si]
         add si, 2
         dec cx
         jnz ciclo1

mov cx, DIM
xor si, si
ciclo2:  pop vettore[si]
         add si, 2
         dec cx
         jnz ciclo2

.exit
end
```

Esercizio 2

- Si scriva un programma che stampi a video il valore decimale di un intero nell'intervallo [0, 2^{16}-1] memorizzato in un'opportuna variabile.

Implementazione

- Si utilizza un algoritmo in due passi:
  - Scomposizione del numero binario nelle sue cifre tramite divisioni successive per 10, salvando i resti e ripetendo l'operazione sul quoziente sino a che questo è diverso da zero
  - Visualizzazione delle cifre così ottenute in ordine inverso a quello di generazione, utilizzando lo stack
  - N.B.: le cifre devono essere convertite in caratteri ASCII prima della stampa.

Codice

```
          .model small
          .stack
          .data

datoin    dw 3721     ; dato da stampare

          .code
          .startup

           mov cx, 10        ; divisore
           mov dx, 0         ; word più significativo del dividendo
           mov ax, datoin    ; word meno significativo del dividendo
           mov bx, 0         ; contatore cifre
```

Codice [cont.]

```
conv:   div cx             ; ax / 10 = ax + dx / 10
        add dx, '0'        ; conversione ASCII
        push dx            ; inserimento in stack di cifra
        mov dx, 0          ; azzeramento dx per divisione successiva
        inc bx             ; incremento contatore cifre
        cmp ax, 0          ; verifica esistenza altre cifre da convertire
        jnz conv

        mov ah, 02h        ; codice system call per stampa carattere
stampa: pop dx             ; prelevamento di cifra da stack (in ordine inverso)
        int 21h            ; system call
        dec bx             ; decremento contatore cifre
        jnz stampa

       .exit
        end
```

Esercizio 3

- Si scriva un programma che richieda all'utente un intero positivo (eventualmente composto da più cifre, e concluso con ENTER) e lo salvi in una variabile di tipo *word*. L'inserimento di valori troppo grandi deve segnalare un errore.
  - Approfondimento: Acquisire 5 interi positivi separati da ENTER e memorizzarli in un vettore di *word*.

Implementazione

- Si utilizza un algoritmo in due passi:
  - nel primo si acquisiscono i caratteri ASCII;
  - nel secondo passo si convertono in intero, valutando la presenza eventuale di overflow
- I due passi possono essere svolti nello stesso ciclo.

Codice

```
DIM EQU 5
LF  EQU 10
CR  EQU 13

.model small
.stack
.data

message db 'Introdurre 5 interi positivi separati da ENTER: ', CR, LF
errore1 db 'ERRORE:   Caratteri non numerici introdotti!!!!!', CR, LF
errore2 db 'ERRORE:   Intero introdotto troppo grande!!!!!!!', CR, LF

vettris dw DIM DUP (?)
fattore dw 10
```

NB: La soluzione proposta include anche la stampa di alcuni messaggi a video allo scopo di rendere più completo il programma, ma che non sono necessari per soddisfare la richiesta.

Codice [cont.]

```
.code
.startup

mov ah, 02h         ; questo blocco stampa un messaggio a video
lea si, message
mov cx, 50          ; lunghezza message
stampa:  mov dl, [si]
    int 21h
    inc si
    loop stampa

mov cx, DIM        ; inizio ciclo acquisizione

lea di, vettris

ciclo:   mov ax, 0
    mov [di], ax
```

Codice [cont.]

```
leggi:   mov ah, 01h  ; codice system call per acquisizione caratteri
   int 21h
   cmp al, CR   ; verifico digitazione “ENTER” (separatore numeri)
   jz next
   cmp al, '0'  ; verifico che il carattere acquisito sia una cifra
   jl err1
   cmp al, '9'
   jg err1
   sub al, '0'  ; conversione cifra ASCII -> binario
   mov ah, 0
   mov bx, ax
   mov ax, [di]
   mul fattore
   jc err2
   add ax, bx
   jc err2
   mov [di], ax
   jmp leggi
next:    add di, 2
   loop ciclo
    jmp fine
```

Codice [cont.]

```
err1:    mov ah, 02h  ; stampa messaggio di errore
lea si, errore1
mov cx, 50          ; lunghezza messaggio
stampa1: mov dl, [si]
   int 21h
   inc si
   loop stampa1
   jmp fine

err2:    mov ah, 02h  ; stampa messaggio di errore
lea si, errore2
mov cx, 50          ; lunghezza messaggio
stampa2: mov dl, [si]
   int 21h
   inc si
   loop stampa2

fine:
.exit
end
```

Esercizio 4

- Scrivere un programma in Assembly che sommi i seguenti numeri rappresentati in un vettore di *byte*: -5, -45, -96, -128
- La somma deve essere salvata nella variabile `risultato` di tipo *doubleword*
- Sommare ancora a `risultato` il valore di `addendo`, variabile di tipo *doubleword* con valore 69000
  - In fase di debug, porre particolare attenzione al modo in cui sono memorizzate le doubleword.

Implementazione

- Le variabili di vettore sono *byte* in CA2
  - Per effettuarne la somma su *word* è necessario estenderne il segno: CBW (NB: solo per CA2)
- Le variabili di tipo *doubleword* sono memorizzate a partire dal byte meno significativo
- Il risultato parziale su *word* deve essere esteso a *doubleword*: CWD (NB: solo per CA2)
- Attenzione alla somma del *carry* quando necessario: istruzione ADC.

Codice

```
.model small
.stack
.data

vettore db -5, -45, -96, -128
risultato dd ?
addendo dd 69000

.code
.startup
mov cx, 4
mov si, 0
xor dx, dx                      ; azzeramento dx
ciclo: mov al, vettore[si]      ; somma elementi vettore in dx
       cbw                      ; estensione del segno: al -> ax
       add dx, ax
       inc si
       loop ciclo
```

Codice [cont.]

```
mov ax, dx
cwd                   ; estensione del segno: ax -> dx, ax

add ax, word ptr addendo        ; notare little-endianness
adc dx, word ptr addendo+2

mov word ptr risultato, ax
mov word ptr risultato+2, dx

.exit
end
```