Esercizi Assembly 8086

M. Rebaudengo – R. Ferrero

Politecnico di Torino
Dipartimento di Automatica e Informatica

Esercizio 1)

Dati in memoria i seguenti due vettori di 50 word ciascuno:

- PREZZI rappresentante i prezzi di 50 articoli venduti in un negozio
- SCONTATI inizialmente di contenuto indeterminato,

si scriva una procedura in linguaggio Assembly 8086 in grado di calcolare il prezzo scontato di ciascun articolo e salvarlo nel corrispondente elemento del vettore SCONTATI. La procedura deve leggere da una variabile intera di tipo word denominata SCONTO l'ammontare dello sconto percentuale da applicare. Si esegua un arrotondamento alla cifra superiore se la parte decimale del prezzo risultante è maggiore o uguale a 0,5.

Inoltre, la procedura deve salvare in una variabile di tipo word TOTSCONTO l'ammontare totale delle riduzioni effettuate.

Esempio:

PREZZI: 39, 1880, 2394, 1000, 1590

SCONTO: 30

SCONTATI: 27, 1316, 1676, 700, 1113

TOTSCONTO: 2071

Di seguito un esempio di programma chiamante:

```
[…]
DIM EQU 5
.DATA
prezzi DW 39, 1880, 2394, 1000, 1590
scontati DW DIM DUP (?)
sconto DW 30
totsconto DW ?

.CODE
.STARTUP
CALL calcola_sconto
[…]
```

Esercizio 2)

Si abbia un vettore contenente alcuni interi rappresentanti anni passati (0 ÷ 2008). Si scriva una **procedura in linguaggio Assembly 8086** che sia in grado di determinare se tali anni sono bisestili. Si ricorda che un anno è bisestile se il suo numero è divisibile per 4, con l'eccezione che gli anni secolari (quelli divisibili per 100) sono bisestili solo se divisibili anche per 400. In altre parole,

```
IF (anno divisibile per 100)
  { IF (anno divisibile per 400)
       Anno_bisestile = TRUE
    ELSE Anno_bisestile = FALSE
  }
ELSE
  {  IF (anno divisibile per 4)
       Anno_bisestile = TRUE
     ELSE Anno_bisestile = FALSE
  }
```

La procedura deve ricevere come input:

- *tramite il registro SI*, l'offset di un vettore di *word* contenente gli anni da valutare
- *tramite il registro DI*, l'offset di un vettore di *byte* della stessa lunghezza, che dovrà contenere, al termine dell'esecuzione della procedura, nelle posizioni corrispondenti agli anni espressi nell'altro vettore, il valore 1 se l'anno è bisestile oppure 0 nel caso opposto
- *tramite il registro BX*, la lunghezza di tali vettori.

Esempio:

anni: 1945, 2008, 1800, 2006, 1748, 1600

risultato: 0, 1, 0, 0, 1, 1

lunghezza: 6

Di seguito un esempio di programma chiamante:

```
[…]
LUNG      equ 6
          .data
anni      dw 1945, 2008, 1800, 2006, 1748, 1600
ris       db LUNG DUP (?)
          .code
          .startup
          lea si, anni
          lea di, ris
          mov bx, lung
          call bisestile
[…]
```

Esercizio 3)

Si scriva una procedura “converti” in linguaggio Assembly 8086 in grado di rimuovere tutte le occorrenze di caratteri ripetuti consecutivamente in una stringa.

Ad esempio, la stringa “notte rossa” (dimensione 11) deve essere trasformata nella stringa “note rosa” (dimensione 9).

La procedura deve ricevere come input tramite *stack*:

- l’indirizzo della stringa origine (tale stringa dovrà essere sovrascritta dalla nuova stringa elaborata)
- la dimensione in byte della stringa origine.

Sempre tramite *stack*, la procedura deve fornire come output la dimensione della stringa trasformata. Non è ammesso l’uso di altre variabili in memoria.

Si supponga dunque che il programma chiamante contenga il seguente codice:

```
[…]
.code
lea ax, stringa
push ax
mov ax, DIMENSIONE
push ax
sub sp, 2
call converti
pop ax
mov DIMENSIONE_AGGIORNATA, ax

add sp, 4
[…]
```

Esercizio 4)

Si scriva una **procedura `potenza`** in linguaggio Assembly 8086 in grado di calcolare l'elevamento a potenza tra interi positivi.

La procedura riceve base ed esponente come *word unsigned* mediante lo stack e restitisce il risultato come *doubleword unsigned*, sempre mediante lo stack.

Di seguito un esempio di programma chiamante:

```
[…]
        .data
risult  DD ?
        .code
        .startup
        PUSH 3       ; base
        PUSH 12      ; esponente
        SUB  SP, 4
        CALL potenza
        POP AX
        POP DX
        ADD SP, 4
        mov risult, AX
        mov risult+2, DX
[…]
```

È inoltre richiesto di verificare la presenza di eventuali condizioni di *overflow*, che devono essere segnalate restituendo il valore esadecimale 0FFFFFFFFh.