Esercizi su Programmazione in Assembler x86 per l'8088

Prof. Riccardo Torlone
Università Roma Tre

Esercizio I

Scrivere un programma in linguaggio assemblativo 8088 che, presi due dati a e b in memoria, calcola l'espressione (a+3)*b ponendo il risultato nel registro accumulatore.

Soluzione Esercizio I

```
!calcolo di (a+3)*b
        _EXIT  = 1
.SECT .TEXT
start:
        MOV    AX,(a)
        ADD    AX,3
        MUL     (b)
        PUSH   0
        PUSH   _EXIT
        SYS
.SECT .DATA
a:      .WORD          5
b:      .WORD   3
```

Esercizio II

Scrivere un programma in linguaggio assemblativo 8088 che, preso un intero n in memoria, calcola la somma dei primi n interi.

Il risultato deve essere stampato sullo standard output (video).

Soluzione Esercizio II

```
!Somma dei primi n numeri
            _EXIT     = 1
            _PRINTF   = 127
.SECT .TEXT
start:
            MOV       AX,0
            MOV       CX,(number)
1:          ADD       AX,CX
            LOOP      1b
            MOV       (result), AX
            PUSH      (result)
            PUSH      (number)
            PUSH      format
            PUSH      _PRINTF
            SYS
            MOV       SP,BP
            PUSH      0
            PUSH      _EXIT
            SYS
.SECT .DATA
number:   .WORD     5
result:     .WORD   1                       !
format: .ASCII "La somma dei primi %d interi e' %d
```

Esercizio III

Scrivere un programma in linguaggio assemblativo 8088 che, preso un numero *a* in memoria, calcola il quadrato del numero facendo uso di una subroutine "square" che ha come unico argomento il numero *a*.

Il risultato deve essere stampato sullo standard output (video).

Soluzione Esercizio III

```
! Calcola la il quadrato di un numero con la subroutine “square”
_EXIT       = 1
_PRINTF     = 127
.SECT .TEXT
start:
            MOV  BP,SP
            PUSH (a)
            CALL square
            MOV  SP,BP
            PUSH AX
            PUSH pfmt
            PUSH _PRINTF
            SYS
            MOV  SP,BP
            PUSH 0
            PUSH _EXIT
            SYS
square:
            PUSH BP
            MOV  BP,SP
            MOV  AX,4(BP)
            MOV  BX,AX
            MUL  BX
            POP  BP
            RET

.SECT .DATA
pfmt: .ASCIZ "Il quadrato e' %d!\n"
a:          .WORD 3
```

Esercizio IV

Scrivere un programma in linguaggio assemblativo 8088 che calcola la somma degli elementi di un vettore vec memorizzato in memoria principale.

Il risultato deve essere stampato sullo standard output (video).

Soluzione Esercizio IV

```
! Stampa la somma di un vettore di interi
            _EXIT       = 1
            _PRINTF     = 127
.SECT .TEXT
start:
            MOV  CX,end-vec
            SHR   CX,1              ! In CX va la lunghezza del vettore
            MOV  BX,vec
            MOV  SI,0
            MOV  AX,0
1:          ADD  AX,(BX)(SI)
            ADD  SI,2
            LOOP 1b
            PUSH AX
            PUSH format
            PUSH _PRINTF
            SYS
            MOV  SP,BP
            PUSH 0
            PUSH _EXIT
            SYS
.SECT .DATA
vec:   .WORD 3,4,7,11,3
end:   .SPACE 1
format: .ASCII "La somma degli elementi del vettore e' %d"
```

Esercizio V

Scrivere un programma in linguaggio assemblativo 8088 che calcola la somma degli elementi di un vettore *vec* memorizzato in memoria principale, facendo uso di una subroutine "vecsum" che ha come argomento la dimensione del vettore e il vettore.

Il risultato deve essere stampato sullo standard output (video).

Soluzione Esercizio V

```
! Stampa la somma di un array di interi mediante una subroutine "vecsum"
                    _EXIT           = 1
                    _PRINTF         = 127
.SECT .TEXT
vecpstrt:
                    MOV  BP,SP
                    PUSH vec
                    MOV  CX,end-vec
                    SHR  CX,1
                    PUSH CX
                    CALL vecsum
                    MOV  SP,BP
                    PUSH AX
                    PUSH format
                    PUSH _PRINTF
                    SYS
                    MOV SP,BP
                    PUSH 0
                    PUSH _EXIT
                    SYS
vecsum:
                    PUSH BP
                    MOV  BP,SP
                    MOV  CX,4(BP)
                    MOV  BX,6(BP)
                    MOV  SI,0
                    MOV  AX,0
1:                  ADD  AX,(BX)(SI)
                    ADD  SI,2
                    LOOP 1b
                    MOV  SP,BP
                    POP  BP
                    RET
.SECT .DATA
vec:   .WORD 3,4,7,11,3
end:   .SPACE 1
format: .ASCII "La somma della stringa e' %d"
.SECT .BSS
```

Esercizio VI

Con riferimento al programma assemblativo 8088 che segue, indicare cosa fa e il valore stampato.

Esercizio VI

```
                _EXIT = 1
                _PRINTF       = 127
.SECT .TEXT
start:
                MOV  CX,num-vec
                SHR  CX,1
                MOV BX,vec
                MOV  SI,0
                MOV  AX,(num)
1:              CMP  AX,(BX)(SI)
                JE   2f
                ADD  SI,2
                LOOP 1b
                MOV  DX,0
                JMP  3f
2:              MOV  DX,1
3:              PUSH DX
                PUSH format
                PUSH _PRINTF
                SYS
                MOV  SP,BP
                PUSH 0
                PUSH _EXIT
                SYS
.SECT .DATA
vec:   .WORD 3,4,7,11,3
num:   .WORD 5
format: .ASCII "%d"
```

Esercizio VIbis

```
! Equivalente al VI utilizzando pero' l'istruzione SCASW insieme alla REPNE
      _EXIT  = 1
      _PRINTF            = 127
.SECT .TEXT
start:
      MOV  CX,num-vec
      SHR  CX,1
      MOV  AX,(num)
      MOV  DI, vec
      CLD
      REPNE SCASW
      JE   1f
      MOV  DX,0
      JMP  2f
1:    MOV  DX,1
2:    PUSH DX
      PUSH format
      PUSH _PRINTF
      SYS
      MOV  SP,BP
      PUSH 0
      PUSH _EXIT
      SYS
.SECT .DATA
vec:    .WORD 3,4,7,11,3
num:    .WORD 11
format: .ASCII "%d"
```

Esercizio VII

Scrivere un programma in linguaggio assemblativo 8088 che verifica se due vettori di interi memorizzati in memoria principale sono identici.

Esercizio VII

```
        _EXIT    = 1
        _PRINTF = 127
.SECT .TEXT
inizio:
        MOV  CX,end1-vec1
        SHR  CX,1
        MOV  AX,end2-vec2
        SHR  AX,1
        CMP  AX,CX
        JNE  1f
        MOV  SI,vec1
        MOV  DI,vec2
        CLD
        REPE CMPSW
        JNE  1f
        PUSH uguali
        JMP  2f
1:      PUSH diversi
2:      PUSH _PRINTF
        SYS
        MOV  SP,BP
        PUSH 0
        PUSH _EXIT
        SYS
.SECT .DATA
vec1: .WORD 3,4,7,11,3
end1: .SPACE 1
vec2: .WORD 3,4,7,11,3
end2: .SPACE 1
uguali: .ASCII "Uguali!\0"
diversi: .ASCII "Diversi!\0"
```

Esercizio VIII

Scrivere un programma in linguaggio assemblativo 8088 che, dato un numero memorizzato in memoria principale, calcola il fattoriale del numero ($n! = n \times (n-1) \times \ldots \times 1$) e lo stampa.

Esercizio VIII

```
!Calcolo del fattoriale
      _EXIT  = 1
      _PRINTF = 127
.SECT .TEXT
start:
      MOV    AX,(number)
      CMP     AX,1
      JG      1f
      MOV     AX,1
      JMP     3f
1:    MOV     CX,AX
      DEC     CX
2:    IMUL    CX
      LOOP    2b
3:    MOV     (result), AX
      PUSH    (result)
      PUSH    (number)
      PUSH     fmt
      PUSH     _PRINTF
      SYS
      MOV     SP,BP
      PUSH    0
      PUSH    _EXIT
      SYS
.SECT .DATA
number: .WORD          5
result: .WORD   1
fmt:.ASCII "il fattoriale di %d e' %d\0"
```

Esercizio IX

Scrivere un programma in linguaggio assemblativo 8088 che dato un numero *n* memorizzato in memoria principale, verifica se è un numero primo.

Consiglio: utilizzare l'istruzione DIV che divide l'argomento per il contenuto di AX mettendo il risultato in AX e il resto in DX

Esercizio IX

```
        _EXIT    = 1
        _PRINTF = 127
.SECT .TEXT
start:
        MOV      BX,(n)
        MOV      CX,BX
1:      DEC      CX
        CMP      CX,1
        JLE      3f
        MOV      AX,BX
        MOV      DX,0
        DIV      CX
        CMP      DX,0
        JE       2f
        JMP      1b
2:      MOV      BX, nonprimo
        JMP      4f
3:      MOV      BX, primo
4:      PUSH     (n)
        PUSH    BX
        PUSH    _PRINTF
        SYS
        MOV      SP,BP
        PUSH     0
        PUSH     _EXIT
        SYS
.SECT .DATA
n:      .WORD   49
primo: .ASCII "%d e' un numero primo\0"
nonprimo: .ASCII "%d non e' un numero primo\0"
```

Esercizio IXbis: altra possibile soluzione

```
        _EXIT  = 1
        _PRINTF = 127
.SECT .TEXT
start:
        MOV    BX,(n)
        MOV    CX,BX
        DEC    CX
1:      CMP    CX,1
        JLE    3f
        MOV    AX,BX
        MOV    DX,0
        IDIV   CX
        CMP    DX,0
        LOOPNZ  1b
2:      MOV    BX, nonprimo
        JMP    4f
3:      MOV    BX, primo
4:      PUSH   (n)
        PUSH    BX
        PUSH    _PRINTF
        SYS
        MOV    SP,BP
        PUSH   0
        PUSH   _EXIT
        SYS
.SECT .DATA
n:      .WORD  49
primo: .ASCII "%d e' un numero primo\0"
nonprimo: .ASCII "%d non e' un numero primo\0"
```

Esercizio X

Scrivere un versione ricorsiva del programma del calcolo del fattoriale.

Esercizio X: Possibile soluzione

```
!Calcolo del fattoriale: versione ricorsiva
        _EXIT = 1
        _PRINTF = 127

.SECT .TEXT
start:
        PUSH  (number)
        CALL   fatt
        POP    CX
        MOV   SP,BP
        PUSH  CX
        PUSH  (number)
        PUSH  fmt
        PUSH  _PRINTF
        SYS
        MOV   SP,BP
        PUSH  0
        PUSH  _EXIT
        SYS

fatt:
        PUSH  BP
        MOV   BP,SP
        MOV   CX,4(BP)
        CMP    CX,1
        JG      1f
        MOV   4(BP),1
        JMP    2f
1:
        DEC   CX
        PUSH  CX
        CALL  fatt
        POP   CX
        MOV   AX,4(BP)
        IMUL  CX
        MOV   4(BP),AX
2:
        MOV   BP,SP
        POP   BP
        RET

.SECT .DATA
number: .WORD 3
fmt: .ASCII "il fattoriale di %d e' %d\n"
```

Esercizio XI

Scrivere un programma in linguaggio assemblativo 8088 che calcola il del prodotto scalare di due vettori (somma dei prodotti degli elementi omologhi). Il programma deve essere dotato di una subrountine **prodvec** avente quattro parametri:

- vec1 (indirizzo del primo vettore)
- vec2 (indirizzo del secondo vettore)
- dimensione (dimensione dei vettori – si assuma che siano della stessa lughezza)
- risultato (parametro di output che al termine dell'esecuzione della subroutine memorizza il risultato del prodotto).

Esercizio XI: possibile soluzione

```
_EXIT       = 1
_PRINTF     = 127
.SECT .TEXT
start:
      MOV  BP,SP
      MOV CX,vec2-vec1
      SHR  CX,1
      PUSH 0       !quarto parametro inz. a zero
      PUSH CX      !terzo parametro
      PUSH vec2    !secondo parametro
      PUSH vec1    !primo parametro
      CALL prodvec
      ADD SP,6     !tolgo i primi tre parametri
      POP AX
      MOV  SP,BP
      PUSH AX
      PUSH fmt
      PUSH _PRINTF
      SYS
      MOV  SP,BP
      PUSH 0
      PUSH _EXIT
      SYS
```

```
prodvec:
      PUSH BP
      MOV  BP,SP
      MOV  CX,8(BP)
      MOV  SI,0
      PUSH 0         !variabile locale inz. a zero
1:    MOV  BX,4(BP)
      MOV  AX,(BX)(SI)
      MOV  BX,6(BP)
      MUL  (BX)(SI)
      ADD  -2(BP),AX
      ADD  SI,2
      LOOP 1b
      POP  10(BP)  !salvo la var. nel 4 argomento
      POP  BP
      RET


.SECT .DATA
vec1: .WORD 3,4,7,11,3
vec2: .WORD 2,6,3,1,0
fmt: .ASCII "Il prodotto dei due vettori e': %d!\0"!
```

Esercizio XII

Scrivere una subroutine PAL in assembler 8088 che, dato una stringa (vettore di caratteri) S memorizzata in memoria principale, stampa restituisce 1 se la stringa S è palindroma (è uguale leggendola nei due versi; per esempio la stringa "anna" è palindroma) e 0 altrimenti.

La subroutine PAL ha come parametri:

- L'indirizzo della stringa da verificare e
- la lunghezza della stringa

Soluzione Esercizio XII

```
! Verifica di stringhe palindrome
_EXIT       = 1
_PRINTF     = 127
.SECT .TEXT
start:
            MOV  BP,SP
            PUSH ends-str  !secondo parametro
            PUSH str !primo parametro
            CALL pal
            MOV  SP,BP
            PUSH AX
            PUSH fmt
            PUSH _PRINTF
            SYS
            MOV  SP,BP
            PUSH 0
            PUSH _EXIT
            SYS
pal:
            PUSH BP
            MOV  BP,SP
            MOV  BX,4(BP)
            MOV  CX,6(BP)
            MOV  SI,CX
            DEC  SI
            MOV  DI,str2
1:          MOVB  AL,(BX)(SI)
            STOSB
            DEC  SI
            LOOP 1b
            MOV  SI,4(BP)
            MOV  DI,str2
            MOV  CX,6(BP)
            REPE CMPSB
            JE   2f
            MOV  AX,0
            JMP  3f
2:          MOV  AX,1
3:          POP  BP
            RET

.SECT .DATA
str: .ASCII "ingegni"
ends: .SPACE 1
fmt: .ASCII "%d"
endf: .SPACE 1
str2: .ASCII "."
```

Esercizio XIII

Scrivere un programma in linguaggio assemblativo 8088 che trova il più grande degli elementi di un vettore vec memorizzato in memoria principale. Si assuma che il vettore abbia almeno un elemento.

Il risultato deve essere stampato sullo standard output (video).

Soluzione Esercizio XIII

```
! Trova il piu' grande tra gli elementi di un vettore di interi
            _EXIT       = 1
            _PRINTF     = 127
.SECT .TEXT
start:
            MOV  CX,end-vec
            SHR  CX,1 !in CX va la dimensione di vec
            MOV  BX,vec !il registro base punta al primo elemento di vec
            MOV  AX,(vec) !inizializzo AX con il primo elemento di vec
1:          CMP  AX,(BX)(SI)
            JGE  2f
            MOV  AX,(BX)(SI)
2:          ADD  SI,2
            LOOP 1b
            PUSH AX
            PUSH format
            PUSH _PRINTF
            SYS
            MOV  SP,BP
            PUSH 0
            PUSH _EXIT
            SYS
.SECT .DATA
vec:    .WORD 3,-4,7,11,34,-4,22,0,5
end:    .SPACE 1
format: .ASCII "Il piu' grande tra elementi del vettore e' %d"
```