SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Architettura degli Elaboratori
Lez. 7 – Esercizi Assembler

Prof. Andrea Sterbini – sterbini@di.uniroma1.it

Argomenti

- **Argomenti della lezione**
  - Soluzione esercizio per casa
  - Esercizi vari di Assembler
- NOTA: i sorgenti dei programmi sono anche allegati alla lezione

Argomenti

- **Argomenti della lezione**
  - Soluzione esercizio per casa
  - Esercizi vari di Assembler
- NOTA: i sorgenti dei programmi sono anche allegati alla lezione

Esercizio (esame del 8-7-13)

Realizzare in assembler, sia in forma ricorsiva che in forma iterativa, la funzione GCD (massimo comun divisore di due interi positivi) definita come segue:

GCD(x, y) = x **se x = y**

GCD(x, y) = GCD(y, x) **se x > y**

GCD(x, y) = GCD(x, y – x) **se x < y**

Esempio di esecuzione

GCD(120, 105) = GCD(105, 120) = GCD(105, 15) = GCD(15, 105) = GCD(15, 90) = GCD(15, 75) = GCD(15, 60) = GCD(15, 45) = GCD(15, 30) = GCD(15, 15) = **15**

GCD: tail recursion optimization

```
# versione ricorsiva
GCD:
    bne $a0,$a1, x_diverso_y
    # GCD(x, x) = x
    move $v0, $a0  # caso base
    jr  $ra
```

GCD: tail recursion optimization

```
# versione ricorsiva
GCD:
    bne $a0,$a1, x_diverso_y
    # GCD(x, x) = x
    move $v0, $a0  # caso base
    jr  $ra
x_diverso_y:   # caso_ricorsivo
    subi $sp, $sp, 4
    sw  $ra, 0($sp)
```

GCD: tail recursion optimization

```
# versione ricorsiva
GCD:
    bne $a0,$a1, x_diverso_y
    # GCD(x, x) = x
    move $v0, $a0  # caso base
    jr  $ra
x_diverso_y:   # caso_ricorsivo
    subi $sp, $sp, 4
    sw  $ra, 0($sp)
    bgt $a0, $a1, x_maggiore_y
```

GCD: tail recursion optimization

```
# versione ricorsiva
GCD:
    bne $a0,$a1, x_diverso_y
    # GCD(x, x) = x
    move $v0, $a0  # caso base
    jr  $ra
x_diverso_y:   # caso_ricorsivo
    subi $sp, $sp, 4
    sw  $ra, 0($sp)
    bgt $a0, $a1, x_maggiore_y
    sub $a1, $a1, $a0
    jal GCD      # GCD(x, y-x)
    j   esci
```

GCD: tail recursion optimization

```
# versione ricorsiva
GCD:
    bne $a0,$a1, x_diverso_y
    # GCD(x, x) = x
    move $v0, $a0  # caso base
    jr  $ra
x_diverso_y:   # caso_ricorsivo
    subi $sp, $sp, 4
    sw  $ra, 0($sp)
    bgt $a0, $a1, x_maggiore_y
    sub $a1, $a1, $a0
    jal GCD      # GCD(x, y-x)
    j   esci




esci:
    lw  $ra, 0($sp)
    addi $sp, $sp, 4
    jr  $ra
```

GCD: tail recursion optimization

```
# versione ricorsiva
GCD:
    bne $a0,$a1, x_diverso_y
    # GCD(x, x) = x
    move $v0, $a0  # caso base
    jr  $ra
x_diverso_y:   # caso_ricorsivo
    subi $sp, $sp, 4
    sw  $ra, 0($sp)
    bgt $a0, $a1, x_maggiore_y
    sub $a1, $a1, $a0
    jal GCD      # GCD(x, y-x)
    j   esci
x_maggiore_y:
    move $v0, $a0
    move $a0, $a1
    move $a1, $v0
    jal GCD      # GCD(y, x)
esci:
    lw  $ra, 0($sp)
    addi $sp, $sp, 4
    jr  $ra
```

GCD: tail recursion optimization

```
# versione ricorsiva
GCD:
    bne $a0,$a1, x_diverso_y
    # GCD(x, x) = x
    move $v0, $a0  # caso base
    jr  $ra
x_diverso_y:   # caso_ricorsivo
    subi $sp, $sp, 4
    sw  $ra, 0($sp)
    bgt $a0, $a1, x_maggiore_y
    sub $a1, $a1, $a0
    jal GCD      # GCD(x, y-x)
    j   esci
x_maggiore_y:
    move $v0, $a0
    move $a0, $a1
    move $a1, $v0
    jal GCD      # GCD(y, x)
esci:
    lw  $ra, 0($sp)
    addi $sp, $sp, 4
    jr  $ra
```

NOTA: non ci sono operazioni dopo la chiamata ricorsiva per cui il secondo ciclo non è necessario

GCD: tail recursion optimization

```
# versione ricorsiva
GCD:
    bne $a0,$a1, x_diverso_y
    # GCD(x, x) = x
    move $v0, $a0  # caso base
    jr  $ra
x_diverso_y:   # caso_ricorsivo
    subi $sp, $sp, 4
    sw  $ra, 0($sp)
    bgt $a0, $a1, x_maggiore_y
    sub $a1, $a1, $a0
    jal GCD      # GCD(x, y-x)
    j   esci
x_maggiore_y:
    move $v0, $a0
    move $a0, $a1
    move $a1, $v0
    jal GCD      # GCD(y, x)
esci:
    lw  $ra, 0($sp)
    addi $sp, $sp, 4
    jr  $ra
```

```
# versione iterativa
GCD:
    bne $a0, $a1, x_diverso_y
    # GCD(x, x) = x
    move $v0, $a0  # caso base
    jr  $ra
```

NOTA: non ci sono operazioni dopo la chiamata ricorsiva per cui il secondo ciclo non è necessario

GCD: tail recursion optimization

```
# versione ricorsiva
GCD:
    bne $a0,$a1, x_diverso_y
    # GCD(x, x) = x
    move $v0, $a0  # caso base
    jr  $ra
x_diverso_y:   # caso_ricorsivo
    subi $sp, $sp, 4
    sw  $ra, 0($sp)
    bgt $a0, $a1, x_maggiore_y
    sub $a1, $a1, $a0
    jal GCD      # GCD(x, y-x)
    j   esci
x_maggiore_y:
    move $v0, $a0
    move $a0, $a1
    move $a1, $v0
    jal GCD      # GCD(y, x)
esci:
    lw  $ra, 0($sp)
    addi $sp, $sp, 4
    jr  $ra
```

```
# versione iterativa
GCD:
    bne $a0, $a1, x_diverso_y
    # GCD(x, x) = x
    move $v0, $a0  # caso base
    jr  $ra
x_diverso_y:


    bgt $a0, $a1, x_maggiore_y
    sub $a1, $a1, $a0
    j   GCD      # GCD(x, y-x)
```

NOTA: non ci sono operazioni dopo la chiamata ricorsiva per cui il secondo ciclo non è necessario

GCD: tail recursion optimization

```
# versione ricorsiva
GCD:
    bne $a0,$a1, x_diverso_y
    # GCD(x, x) = x
    move $v0, $a0  # caso base
    jr  $ra
x_diverso_y:   # caso_ricorsivo
    subi $sp, $sp, 4
    sw  $ra, 0($sp)
    bgt $a0, $a1, x_maggiore_y
    sub $a1, $a1, $a0
    jal GCD      # GCD(x, y-x)
    j   esci
x_maggiore_y:
    move $v0, $a0
    move $a0, $a1
    move $a1, $v0
    jal GCD      # GCD(y, x)
esci:
    lw  $ra, 0($sp)
    addi $sp, $sp, 4
    jr  $ra
```

```
# versione iterativa
GCD:
    bne $a0, $a1, x_diverso_y
    # GCD(x, x) = x
    move $v0, $a0  # caso base
    jr  $ra
x_diverso_y:


    bgt $a0, $a1, x_maggiore_y
    sub $a1, $a1, $a0
    j   GCD      # GCD(x, y-x)

x_maggiore_y:
    move $v0, $a0
    move $a0, $a1
    move $a1, $v0
    j   GCD      # GCD(y, x)
```

NOTA: non ci sono operazioni dopo la chiamata ricorsiva per cui il secondo ciclo non è necessario

Raggruppalettere (esame 21-7-14)

- Si realizzi la funzione RICORSIVA **raggruppalettere** che raggruppa le lettere di una stringa mettendo:
- - prima le lettere (ovvero i caratteri compresi tra 'a' e 'z' e tra 'A' e 'Z')
- - poi i restanti caratteri (tutto il testo comprese lettere accentate, cifre, interpunzione o altro)
- **NOTA:** l'ordine dei caratteri in ciascuno dei due gruppi non è importante
- **Argomenti da passare (o altro se preferite):**

-indirizzo della stringa

-suo numero di caratteri

- **Risultato da tornare:**

-nessuno (la stringa va modificata in memoria)

Raggruppalettere (esame 21-7-14)

- Si realizzi la funzione RICORSIVA **raggruppalettere** che raggruppa le lettere di una stringa mettendo:
- - prima le lettere (ovvero i caratteri compresi tra 'a' e 'z' e tra 'A' e 'Z')
- - poi i restanti caratteri (tutto il testo comprese lettere accentate, cifre, interpunzione o altro)
- **NOTA:** l'ordine dei caratteri in ciascuno dei due gruppi non è importante
- **Argomenti da passare (o altro se preferite):**
  - indirizzo della stringa
  - suo numero di caratteri
- **Risultato da tornare:**
  - nessuno (la stringa va modificata in memoria)

- **Esempio di algoritmo ricorsivo:**
  - se la stringa è lunga 1 o 0 caratteri torna senza fare nulla (è già ordinata)

Raggruppalettere (esame 21-7-14)

- Si realizzi la funzione RICORSIVA **raggruppalettere** che raggruppa le lettere di una stringa mettendo:
- - prima le lettere (ovvero i caratteri compresi tra 'a' e 'z' e tra 'A' e 'Z')
- - poi i restanti caratteri (tutto il testo comprese lettere accentate, cifre, interpunzione o altro)
- **NOTA:** l'ordine dei caratteri in ciascuno dei due gruppi non è importante
- **Argomenti da passare (o altro se preferite):**
  - indirizzo della stringa
  - suo numero di caratteri
- **Risultato da tornare:**
  - nessuno (la stringa va modificata in memoria)

- **Esempio di algoritmo ricorsivo:**
  - se la stringa è lunga 1 o 0 caratteri torna senza fare nulla (è già ordinata)
  - se il primo carattere è una lettera esegui raggruppalettere sul resto della stringa (la prima lettera è già al suo posto)

Raggruppalettere (esame 21-7-14)

- Si realizzi la funzione RICORSIVA **raggruppalettere** che raggruppa le lettere di una stringa mettendo:
- - prima le lettere (ovvero i caratteri compresi tra 'a' e 'z' e tra 'A' e 'Z')
- - poi i restanti caratteri (tutto il testo comprese lettere accentate, cifre, interpunzione o altro)
- **NOTA:** l'ordine dei caratteri in ciascuno dei due gruppi non è importante
- **Argomenti da passare (o altro se preferite):**
  - indirizzo della stringa
  - suo numero di caratteri
- **Risultato da tornare:**
  - nessuno (la stringa va modificata in memoria)

- **Esempio di algoritmo ricorsivo:**
  - se la stringa è lunga 1 o 0 caratteri torna senza fare nulla (è già ordinata)
  - se il primo carattere è una lettera esegui raggruppalettere sul resto della stringa (la prima lettera è già al suo posto)
  - se il primo carattere non è una lettera scambiala con l'ultima ed esegui raggruppalettere sul resto della stringa più corta di un carattere (l'ultima è stata messa a posto, e del primo non sappiamo nulla)

Raggruppalettere (esame 21-7-14)

- Si realizzi la funzione RICORSIVA **raggruppalettere** che raggruppa le lettere di una stringa mettendo:
- - prima le lettere (ovvero i caratteri compresi tra 'a' e 'z' e tra 'A' e 'Z')
- - poi i restanti caratteri (tutto il testo comprese lettere accentate, cifre, interpunzione o altro)
- **NOTA:** l'ordine dei caratteri in ciascuno dei due gruppi non è importante
- **Argomenti da passare (o altro se preferite):**
  - indirizzo della stringa
  - suo numero di caratteri
- **Risultato da tornare:**
  - nessuno (la stringa va modificata in memoria)

- **Esempio di algoritmo ricorsivo:**
  - se la stringa è lunga 1 o 0 caratteri torna senza fare nulla (è già ordinata)
  - se il primo carattere è una lettera esegui raggruppalettere sul resto della stringa (la prima lettera è già al suo posto)
  - se il primo carattere non è una lettera scambiala con l'ultima ed esegui raggruppalettere sul resto della stringa più corta di un carattere (l'ultima è stata messa a posto, e del primo non sappiamo nulla)
- **Esempio di algoritmo iterativo:**

Raggruppalettere (esame 21-7-14)

- Si realizzi la funzione RICORSIVA **raggruppalettere** che raggruppa le lettere di una stringa mettendo:
- - prima le lettere (ovvero i caratteri compresi tra 'a' e 'z' e tra 'A' e 'Z')
- - poi i restanti caratteri (tutto il testo comprese lettere accentate, cifre, interpunzione o altro)
- **NOTA:** l'ordine dei caratteri in ciascuno dei due gruppi non è importante
- **Argomenti da passare (o altro se preferite):**
  - indirizzo della stringa
  - suo numero di caratteri
- **Risultato da tornare:**
  - nessuno (la stringa va modificata in memoria)

- **Esempio di algoritmo ricorsivo:**
  - se la stringa è lunga 1 o 0 caratteri torna senza fare nulla (è già ordinata)
  - se il primo carattere è una lettera esegui raggruppalettere sul resto della stringa (la prima lettera è già al suo posto)
  - se il primo carattere non è una lettera scambiala con l'ultima ed esegui raggruppalettere sul resto della stringa più corta di un carattere (l'ultima è stata messa a posto, e del primo non sappiamo nulla)
- **Esempio di algoritmo iterativo:**
  - Scandisco i caratteri e sposto in fondo quelli che non sono lettere

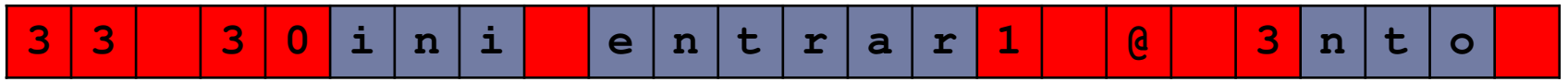
Raggruppalettere (esame 21-7-14)

- Si realizzi la funzione RICORSIVA **raggruppalettere** che raggruppa le lettere di una stringa mettendo:
- - prima le lettere (ovvero i caratteri compresi tra 'a' e 'z' e tra 'A' e 'Z')
- - poi i restanti caratteri (tutto il testo comprese lettere accentate, cifre, interpunzione o altro)
- **NOTA:** l'ordine dei caratteri in ciascuno dei due gruppi non è importante
- **Argomenti da passare (o altro se preferite):**
  - indirizzo della stringa
  - suo numero di caratteri
- **Risultato da tornare:**
  - nessuno (la stringa va modificata in memoria)

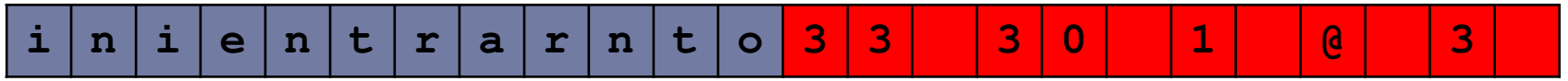
- **Esempio di algoritmo ricorsivo:**
  - se la stringa è lunga 1 o 0 caratteri torna senza fare nulla (è già ordinata)
  - se il primo carattere è una lettera esegui raggruppalettere sul resto della stringa (la prima lettera è già al suo posto)
  - se il primo carattere non è una lettera scambiala con l'ultima ed esegui raggruppalettere sul resto della stringa più corta di un carattere (l'ultima è stata messa a posto, e del primo non sappiamo nulla)
- **Esempio di algoritmo iterativo:**
  - Scandisco i caratteri e sposto in fondo quelli che non sono lettere
  - (servono due indici a inizio e fine)

Esempio

- Esempio di input

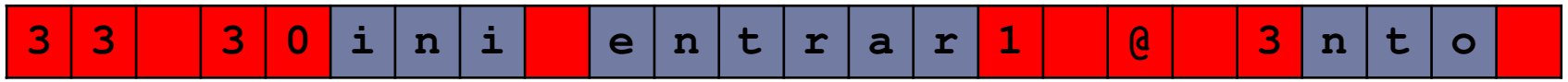


- Esempio di output (anche altri ordinamenti sono validi)

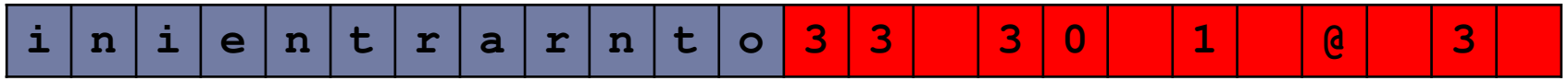


Esempio

- Esempio di input



- Esempio di output (anche altri ordinamenti sono validi)




```
.text
is_lettera:     # registri modificati: $t0..$t3 e $ra
   sge  $t0, $a0, 'a'       # se il carattere è compreso tra `a'
   sle  $t1, $a0, 'z'       # e `z' è una minuscola
   sge  $t2, $a0, 'A'       # oppure tra `A'
   sle  $t3, $a0, 'Z'       # e `Z' è maiuscola
   and  $t0, $t0, $t1       # se è minuscola
   and  $t1, $t2, $t3       # oppure maiuscola
   or   $v0, $t0, $t1       # allora è una lettera
   jr   $ra
```

Versione iterativa

```
raggruppalettere: # $a0, $a1 indici del primo e ultimo carattere
   blt   $a0, $a1, ancora     # ancora caratteri da esaminare?
   jr    $ra
```

Versione iterativa

```
raggruppalettere: # $a0, $a1 indici del primo e ultimo carattere
   blt   $a0, $a1, ancora     # ancora caratteri da esaminare?
   jr    $ra
ancora:
```

Versione iterativa

```
raggruppalettere: # $a0, $a1 indici del primo e ultimo carattere
   blt   $a0, $a1, ancora     # ancora caratteri da esaminare?
   jr    $ra
ancora:
   # qui devo preservare $a0 e $ra su stack
   lb    $a0, stringa($a0)    # leggo il primo carattere
   jal   is_lettera           # vedo se è una lettera
   # qui devo ripristinare $a0 e $ra da stack
```

Versione iterativa

```
raggruppalettere: # $a0, $a1 indici del primo e ultimo carattere
   blt   $a0, $a1, ancora     # ancora caratteri da esaminare?
   jr    $ra
ancora:
   # qui devo preservare $a0 e $ra su stack
   lb    $a0, stringa($a0)    # leggo il primo carattere
   jal   is_lettera           # vedo se è una lettera
   # qui devo ripristinare $a0 e $ra da stack
   beqz  $v0, scambia         # se non lo è lo scambio con l'ultimo
```

Versione iterativa

```
raggruppalettere: # $a0, $a1 indici del primo e ultimo carattere
   blt   $a0, $a1, ancora     # ancora caratteri da esaminare?
   jr    $ra
ancora:
   # qui devo preservare $a0 e $ra su stack
   lb    $a0, stringa($a0)    # leggo il primo carattere
   jal   is_lettera           # vedo se è una lettera
   # qui devo ripristinare $a0 e $ra da stack
   beqz  $v0, scambia         # se non lo è lo scambio con l'ultimo
   addi  $a0, $a0, 1          # la prima è una lettera x cui avanzo
   j     raggruppalettere     # continuo
```

Versione iterativa

```
raggruppalettere: # $a0, $a1 indici del primo e ultimo carattere
   blt   $a0, $a1, ancora     # ancora caratteri da esaminare?
   jr    $ra
ancora:
   # qui devo preservare $a0 e $ra su stack
   lb    $a0, stringa($a0)    # leggo il primo carattere
   jal   is_lettera           # vedo se è una lettera
   # qui devo ripristinare $a0 e $ra da stack
   beqz  $v0, scambia         # se non lo è lo scambio con l'ultimo
   addi  $a0, $a0, 1          # la prima è una lettera x cui avanzo
   j     raggruppalettere     # continuo
scambia:
```

Versione iterativa

```
raggruppalettere: # $a0, $a1 indici del primo e ultimo carattere
   blt   $a0, $a1, ancora     # ancora caratteri da esaminare?
   jr    $ra
ancora:
   # qui devo preservare $a0 e $ra su stack
   lb    $a0, stringa($a0)    # leggo il primo carattere
   jal   is_lettera           # vedo se è una lettera
   # qui devo ripristinare $a0 e $ra da stack
   beqz  $v0, scambia         # se non lo è lo scambio con l'ultimo
   addi  $a0, $a0, 1          # la prima è una lettera x cui avanzo
   j     raggruppalettere     # continuo
scambia:
   lb    $t0, stringa($a0)    # leggo il primo carattere
   lb    $t1, stringa($a1)    # leggo l'ultimo carattere
   sb    $t0, stringa($a1)    # lo sostituisco col primo
   sb    $t1, stringa($a0)    # sostituisco il secondo al I°
```

Versione iterativa

```
raggruppalettere: # $a0, $a1 indici del primo e ultimo carattere
   blt   $a0, $a1, ancora     # ancora caratteri da esaminare?
   jr    $ra
ancora:
   # qui devo preservare $a0 e $ra su stack
   lb    $a0, stringa($a0)    # leggo il primo carattere
   jal   is_lettera           # vedo se è una lettera
   # qui devo ripristinare $a0 e $ra da stack
   beqz  $v0, scambia         # se non lo è lo scambio con l'ultimo
   addi  $a0, $a0, 1          # la prima è una lettera x cui avanzo
   j     raggruppalettere     # continuo
scambia:
   lb    $t0, stringa($a0)    # leggo il primo carattere
   lb    $t1, stringa($a1)    # leggo l'ultimo carattere
   sb    $t0, stringa($a1)    # lo sostituisco col primo
   sb    $t1, stringa($a0)    # sostituisco il secondo al I°
   subi  $a1, $a1, 1          # diminuisco il secondo indice
   j     raggruppalettere     # continuo
```

Versione ricorsiva

Versione ricorsiva

```
raggruppalettere:
   # $a0, $a1 indici del primo e ultimo carattere
   blt   $a0, $a1, caso_ric   # non sono finiti i caratteri?
   jr    $ra                  # caso base, nulla da fare
```

Versione ricorsiva

```
raggruppalettere:
   # $a0, $a1 indici del primo e ultimo carattere
   blt   $a0, $a1, caso_ric   # non sono finiti i caratteri?
   jr    $ra                  # caso base, nulla da fare

caso_ric:
   # qui devo preservare $a0 e $ra su stack
   lb    $a0, stringa($a0)    # leggo il primo carattere
   jal   is_lettera           # vedo se è una lettera
   lw    $a0, 4($sp)          # recupero $a0 che serve dopo
   beqz  $v0, scambia         # se non lo è scambio con l'ultimo
   addi  $a0, $a0, 1          # la prima è una lettera e avanzo
   jal   raggruppalettere     # continuo con chiamata ricorsiva
   # qui devo ripristinare $a0 e $ra da stack
   jr    $ra                  # caso base, nulla da fare
```

Secondo caso ricorsivo

```
scambia:
   # scambio primo e ultimo carattere (che va nella parte
delle NON-lettere)
   lb    $t0, stringa($a0)  # leggo il primo carattere
   lb    $t1, stringa($a1)  # leggo l'ultimo carattere
   sb    $t0, stringa($a1)  # lo sostituisco col primo
   sb    $t1, stringa($a0)  # sostituisco il secondo al I°
```

Secondo caso ricorsivo

```
scambia:
   # scambio primo e ultimo carattere (che va nella parte
delle NON-lettere)
   lb    $t0, stringa($a0)  # leggo il primo carattere
   lb    $t1, stringa($a1)  # leggo l'ultimo carattere
   sb    $t0, stringa($a1)  # lo sostituisco col primo
   sb    $t1, stringa($a0)  # sostituisco il secondo al I°
   # e vado avanti a risolvere il resto della stringa
   subi  $a1, $a1, 1        # diminuisco il secondo indice
   jal   raggruppalettere   # continuo
   #qui devo ripristinare $ra e $a0 da stack
   jr    $ra
```

Secondo caso ricorsivo

```
scambia:
   # scambio primo e ultimo carattere (che va nella parte
delle NON-lettere)
   lb    $t0, stringa($a0)  # leggo il primo carattere
   lb    $t1, stringa($a1)  # leggo l'ultimo carattere
   sb    $t0, stringa($a1)  # lo sostituisco col primo
   sb    $t1, stringa($a0)  # sostituisco il secondo al I°
   # e vado avanti a risolvere il resto della stringa
   subi  $a1, $a1, 1        # diminuisco il secondo indice
   jal   raggruppalettere   # continuo
   #qui devo ripristinare $ra e $a0 da stack
   jr    $ra
```

NOTA: anche in questo caso non ci sono istruzioni dopo la ricorsione quindi è possibile usare l'ottimizzazione della ricorsione di coda per eliminare la ricorsione facilmente

ContaLettere (esame 21-7-14)

- Si realizzi la funzione RICORSIVA **contaLettere** che conta il numero di lettere, cifre ed altri caratteri nella stringa:
- **Caso base:** se la stringa è finita (il carattere corrente è 0) torna la tripla <0, 0, 0>
- **Caso ricorsivo:** altrimenti chiama contaLettere sul resto della stringa e:

a) se la prima lettera è una lettera (tra 'a' e 'z' e tra 'A' e 'Z') somma 1 al primo valore tornato

b) se la prima lettera è una cifra (tra '0' e '9') somma 1 al secondo valore tornato

c) altrimenti somma 1 al terzo valore tornato

- **Argomenti da passare:**

-indirizzo della stringa

-suo numero di caratteri

- **Risultato da tornare:**

-la terna < #lettere, #cifre, #altro >

- **NOTA:** per semplicità per tornare i 3 valori usiamo 3 registri $v0, $v1, $s0 invece che lo stack

Versione ricorsiva: caso base

```
is_between:
   # torna $s1=1 se $a1 è nell'intervallo [$a2, $a3]
   li    $s1, 0          # NO (0)
   blt   $a1, $a2, NO    # se $a0 è minore di $a1 → 0
   bgt   $a1, $a3, NO    # se $a0 è maggiore di $a2 → 0
   li    $s1, 1          # altrimenti SI (1)
NO: jr   $ra              # registri modificati: $s1 e $ra
```

Versione ricorsiva: caso base

```
is_between:
   # torna $s1=1 se $a1 è nell'intervallo [$a2, $a3]
   li    $s1, 0          # NO (0)
   blt   $a1, $a2, NO    # se $a0 è minore di $a1 → 0
   bgt   $a1, $a3, NO    # se $a0 è maggiore di $a2 → 0
   li    $s1, 1          # altrimenti SI (1)
NO: jr   $ra              # registri modificati: $s1 e $ra
```

Versione ricorsiva: caso base

```
is_between:
   # torna $s1=1 se $a1 è nell'intervallo [$a2, $a3]
   li    $s1, 0          # NO (0)
   blt   $a1, $a2, NO    # se $a0 è minore di $a1 → 0
   bgt   $a1, $a3, NO    # se $a0 è maggiore di $a2 → 0
   li    $s1, 1          # altrimenti SI (1)
NO: jr   $ra              # registri modificati: $s1 e $ra
```

Versione ricorsiva: caso base

```
is_between:
   # torna $s1=1 se $a1 è nell'intervallo [$a2, $a3]
   li    $s1, 0          # NO (0)
   blt   $a1, $a2, NO    # se $a0 è minore di $a1 → 0
   bgt   $a1, $a3, NO    # se $a0 è maggiore di $a2 → 0
   li    $s1, 1          # altrimenti SI (1)
NO: jr   $ra              # registri modificati: $s1 e $ra
```

Contalettere: caso ricorsivo

```
ancora:
   addi   $a0, $a0, 1          # vado avanti di un carattere
   jal    contalettere         # conto il resto delle lettere
```

Contalettere: caso ricorsivo

```
ancora:
   addi  $a0, $a0, 1          # vado avanti di un carattere
   jal   contalettere         # conto il resto delle lettere
   # analizzo il primo carattere ($a1) per sapere dove sommare 1
   li    $a2, 'a'             # estremo inferiore
   li    $a3, `z'             # estremo superiore
   jal   is_between           # è una lettera minuscola?
```

Contalettere: caso ricorsivo

```
ancora:
   addi  $a0, $a0, 1          # vado avanti di un carattere
   jal   contalettere         # conto il resto delle lettere
   # analizzo il primo carattere ($a1) per sapere dove sommare 1
   li    $a2, 'a'             # estremo inferiore
   li    $a3, `z'             # estremo superiore
   jal   is_between           # è una lettera minuscola?
   beqz  $s1, non_minuscola   # se non lo è vado avanti
   addi  $v0, $v0, 1          # se lo è sommo 1 al n° di lettere
   j     fine_ricorsione      # i ripristini sono tutti assieme
```

Contalettere: caso ricorsivo

```
ancora:
   addi  $a0, $a0, 1          # vado avanti di un carattere
   jal   contalettere         # conto il resto delle lettere
   # analizzo il primo carattere ($a1) per sapere dove sommare 1
   li    $a2, 'a'             # estremo inferiore
   li    $a3, `z'             # estremo superiore
   jal   is_between           # è una lettera minuscola?
   beqz  $s1, non_minuscola   # se non lo è vado avanti
   addi  $v0, $v0, 1          # se lo è sommo 1 al n° di lettere
   j     fine_ricorsione      # i ripristini sono tutti assieme

non_minuscola:
   li    $a2, `A'             # estremo inferiore
   li    $a3, `Z'             # estremo superiore
   jal   is_between           # è una lettera maiuscola?
```

Contalettere: caso ricorsivo

```
ancora:
   addi  $a0, $a0, 1          # vado avanti di un carattere
   jal   contalettere         # conto il resto delle lettere
   # analizzo il primo carattere ($a1) per sapere dove sommare 1
   li    $a2, 'a'             # estremo inferiore
   li    $a3, 'z'             # estremo superiore
   jal   is_between           # è una lettera minuscola?
   beqz  $s1, non_minuscola   # se non lo è vado avanti
   addi  $v0, $v0, 1          # se lo è sommo 1 al n° di lettere
   j     fine_ricorsione      # i ripristini sono tutti assieme

non_minuscola:
   li    $a2, 'A'             # estremo inferiore
   li    $a3, 'Z'             # estremo superiore
   jal   is_between           # è una lettera maiuscola?
   beqz  $s1, non_lettera     # se non lo è vado avanti
   addi  $v0, $v0, 1          # se lo è sommo 1 al n° di lettere
   j     fine_ricorsione      # metto i ripristini tutti assieme
```

Contalettere: fine caso ricorsivo

Contalettere: fine caso ricorsivo

```
non_lettera:
   li    $a2, '0'            # estremo inferiore
   li    $a3, '9'            # estremo superiore
   jal   is_between          # è una cifra?
```

Contalettere: fine caso ricorsivo

```
non_lettera:
   li    $a2, '0'            # estremo inferiore
   li    $a3, '9'            # estremo superiore
   jal   is_between          # è una cifra?
   beqz  $s1, altro          # se non lo è vado avanti
   addi  $v1, $v1, 1         # se lo è sommo 1 al n° di cifre
   j     fine_ricorsione     # i ripristini sono tutti assieme
```

Contalettere: fine caso ricorsivo

```
non_lettera:
   li    $a2, '0'            # estremo inferiore
   li    $a3, '9'            # estremo superiore
   jal   is_between          # è una cifra?
   beqz  $s1, altro          # se non lo è vado avanti
   addi  $v1, $v1, 1         # se lo è sommo 1 al n° di cifre
   j     fine_ricorsione     # i ripristini sono tutti assieme

altro:
   addi  $s0, $s0, 1         # sommo 1 al n° di altre lettere
```

Contalettere: fine caso ricorsivo

```
non_lettera:
   li    $a2, '0'            # estremo inferiore
   li    $a3, '9'            # estremo superiore
   jal   is_between          # è una cifra?
   beqz  $s1, altro          # se non lo è vado avanti
   addi  $v1, $v1, 1         # se lo è sommo 1 al n° di cifre
   j     fine_ricorsione     # i ripristini sono tutti assieme

altro:
   addi  $s0, $s0, 1         # sommo 1 al n° di altre lettere

fine_ricorsione:
   lw    $a1, 0($sp)         # ripristino $a0
   lw    $ra, 4($sp)         # ripristino $ra
   addi  $sp, $sp, 8         # disalloco le 2 word
   jr    $ra
```