



# **Architettura degli Elaboratori**

## **Lez. 7 – Esercizi Assembler**

Prof. Andrea Sterbini – [sterbini@di.uniroma1.it](mailto:sterbini@di.uniroma1.it)



## Argomenti della lezione

- Soluzione esercizio per casa
- Esercizi vari di Assembler

NOTA: i sorgenti dei programmi sono anche allegati alla lezione

### Esercizio per casa (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:

$$\text{GCD}(x, y) = x \quad \text{se } x = y$$

$$\text{GCD}(x, y) = \text{GCD}(y, x) \quad \text{se } x > y$$

$$\text{GCD}(x, y) = \text{GCD}(x, y - x) \quad \text{se } x < y$$

Esempio di esecuzione

$$\begin{aligned} \text{GCD}(120, 105) &= \text{GCD}(105, 120) = \text{GCD}(105, 15) = \text{GCD}(15, 105) = \\ \text{GCD}(15, 90) &= \text{GCD}(15, 75) = \text{GCD}(15, 60) = \text{GCD}(15, 45) = \text{GCD}(15, 30) = \\ &= \text{GCD}(15, 15) = \mathbf{15} \end{aligned}$$

# GCD: tail recursion optimization



SAPIENZA  
UNIVERSITÀ DI ROMA  
DIPARTIMENTO DI INFORMATICA

# 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

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

|   |   |  |   |   |   |   |   |  |   |   |   |   |   |   |   |  |   |  |   |   |   |   |  |
|---|---|--|---|---|---|---|---|--|---|---|---|---|---|---|---|--|---|--|---|---|---|---|--|
| 3 | 3 |  | 3 | 0 | i | n | i |  | e | n | t | r | a | r | 1 |  | @ |  | 3 | n | t | o |  |
|---|---|--|---|---|---|---|---|--|---|---|---|---|---|---|---|--|---|--|---|---|---|---|--|

Esempio di output (anche altri ordinamenti sono validi)

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |   |   |  |   |  |   |  |   |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|---|---|--|---|--|---|--|---|--|
| i | n | i | e | n | t | r | a | r | n | t | o | 3 | 3 |  | 3 | 0 |  | 1 |  | @ |  | 3 |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|---|---|--|---|--|---|--|---|--|

**.text**

```
is_letera:      # 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
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 1°
    subi $a1, $a1, 1    # diminuisco il secondo indice
    j    raggruppalettere # continuo
```

# 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 1°  
# 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)



SAPIENZA  
UNIVERSITÀ DI ROMA  
DIPARTIMENTO DI INFORMATICA

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:

- se la prima lettera è una lettera (tra 'a' e 'z' e tra 'A' e 'Z') somma 1 al primo valore tornato
- se la prima lettera è una cifra (tra '0' e '9') somma 1 al secondo valore tornato
- 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

*contalettere:* # \$a0 puntatore al primo carattere della stringa

subi \$sp, \$sp, 8 # alloco 2 word x preservare \$a0 e \$ra

sw \$a1, 0(\$sp) # salvo \$a1 che contiene il carattere

sw \$ra, 4(\$sp) # salvo \$ra

lb \$a1, (\$a0) # leggo il primo carattere

bnez \$a1, ancora # se non è 0 fine str → casi ricorsivi

li \$v0, 0 # numero di lettere

li \$v1, 0 # numero di cifre

li \$s0, 0 # numero di altri caratteri

j fine\_ricorsione # metto i ripristini 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?
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



SAPIENZA  
UNIVERSITÀ DI ROMA  
DIPARTIMENTO DI INFORMATICA

*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
```