



SAPIENZA
UNIVERSITÀ DI ROMA

ESERCIZI STACK

Dott. Franco Liberati

Argomenti

01

ESERCIZI STACK



The background of the slide is a dark blue, high-tech illustration. It features a central, glowing blue microchip or processor mounted on a circuit board. Numerous glowing blue lines, resembling circuit traces or data paths, extend from the chip and across the frame. Small, glowing blue dots are scattered along these lines and in the background, creating a sense of digital connectivity and data flow.

Esercizi Stack



PER GLI STUDENTI



STUDIARE OGNI PROGRAMMA

E

ANALIZZARE L'EVOLUZIONE DELLO STACK
(ANCHE CON L'USO DI DIGRAMMI E SCHEMI)



Esercizio Stack



Si consideri la funzione f definita su interi

$$f(x) = f(x-2) - 2$$

$$f(1) = 14$$

$$f(0) = 10$$

Si realizzi un programma in assembler MIPS che, definito un intero positivo $x \geq 2$, calcola il corrispondente valore di $f(x)$ in modo **ricorsivo**

ESEMPIO:

$$f(6) = f(4) - 2 = (f(2) - 2) - 2 = ((f(0) - 2) - 2) - 2 = 10 - 2 - 2 - 2 = 4$$

$$f(5) = f(3) - 2 = f(1) - 2 - 2 = 14 - 2 - 2 = 10$$



Esercizio stack

Soluzione



.text

.globl main

main:

li \$v0,5 #lettura valore di ingresso

syscall

move \$a0,\$v0 #spostamento di X in registro preservante

jal Funzione #salto a funzione ricorsiva

move \$a0,\$v0 #recupero del valore di ritorno della funzione ricorsiva

li \$v0,1 #stampa del risultato

syscall

li \$v0, 10 #terminazione programma

syscall

Esercizio stack

Soluzione

Funzione:

```
li $t1,2  
blt $a0, $t1, caso_base  
subu $sp, $sp, 8
```

```
sw $a0, 0($sp)  
sw $ra, 4($sp)  
sub $a0, $a0, 2  
jal Funzione  
lw $a0, 0($sp)  
lw $ra, 4($sp)  
addi $sp, $sp, 8  
sub $v0, $v0, 2  
jr $ra
```

caso_base:

```
li $v0, 10  
beqz $a0, salta  
li $v0, 14
```

salta:

```
jr $ra
```

#confronto per stabilire se il valore analizzato fa riferimento al caso base

#salto al caso base

**#PUSH del valore X e dell'indirizzo di ritorno (nel primo caso al main; negli altri casi
#a dopo la chiamata ricorsiva)**

#

#

#aggiornamento del valore di X (x-2)

#POP dei valori precedentemente custoditi nello stack

#

#

#costante da sottrarre in base all'elemento selezionato

#salto per il POP successivo o per il ritorno del main

#caso base X=0

#caso base X=1

#salto per il POP o per il ritorno del main



Esercizio Stack



Si consideri la funzione f definita su interi

$$f(x,y) = 2 * f(x-2,y-5)$$

$$f(0,y) = 1$$

$$f(x,0) = 2$$

$$f(0,0) = 3$$

Si realizzi un programma in assembler MIPS che, definiti due interi positivi $x \geq 2$ e $y \geq 2$, calcola il corrispondente valore di $f(x,y)$ in modo **ricorsivo**

ESEMPIO:

$$f(4,25) = 2 * f(2,20) = 2 * 2 * f(0,15) = 2 * 2 * 1 = 2 * 2 * 1 = 4$$

$$f(8,10) = 2 * f(6,5) = 2 * 2 * f(4,0) = 2 * 2 * 2 = 8$$

$$f(4,10) = 2 * f(2,5) = 2 * 2 * f(0,0) = 2 * 2 * 3 = 12$$



Esercizio stack

Soluzione



main:

```
.text
.globl main

li $v0,5          #lettura vprimo alore di ingresso
syscall
move $a0,$v0      #spostamento di X in registro preservante
li $v0,5          #lettura secondo valore di ingresso
syscall
move $a1,$v0      #spostamento di Y in registro preservante
jal Funzione      #salto a funzione ricorsiva
move $a0,$v0      #recupero del valore di ritorno della funzione ricorsiva
li $v0,1          #stampa del risultato
syscall

li $v0, 10        #terminazione programma
syscall
```

Esercizio stack

Soluzione

Funzione:

```
beqz $a0,caso_base
beqz $a1,caso_base
```

#confronto per stabilire il caso base
#

```
subu $sp, $sp, 8
sw $ra, 0($sp)
sub $a0,$a0,2
sub $a1,$a1,5
jal Funzione
lw $ra, 0($sp)
addi $sp, $sp, 8
mul $v0,$v0,2
jr $ra
```

#PUSH del valore X e dell'indirizzo di ritorno (nel primo caso al main; negli altri casi a dopo la chiamata ricorsiva)

#aggiornamento X (x-2)
#aggiornamento Y (y-5)

#POP dei valori di RA precedentemente custoditi nello stack

#costante da moltiplicare all'elemento selezionato
#salto per il POP successivo o per il ritorno del main

caso_base:

```
bnez $a0,zero2
bnez $a1,zero1
li $v0,3
j finecasi
```

#x=0e y=0

zero1:

```
li $v0,1
j finecasi
```

zero2:

```
li $v0,2
j finecasi
```

#non serve, ma lo scrivo per uniformità

finecasi:

```
jr $ra
```



Esercizio Stack



Si consideri la funzione f definita su interi

$$f(x,y) = (x-y) * f(x-5,y+3)$$

$$f(x,y) = 4 \text{ se } x \leq 0$$

Si realizzi un programma in assembler MIPS che, definiti due interi positivi e $x \geq 5$, calcola il corrispondente valore di $f(x,y)$ in modo **ricorsivo**

ESEMPIO:

$$f(15,4) = 11 * f(10,7) = 11 * 3 * f(5,10) = 11 * 3 * -5 * f(0,13) = 11 * 3 * -5 * 4 = -660$$



Esercizio stack

Soluzione



main:

```
.text
.globl main

li $v0,5          #lettura vprimo alore di ingresso
syscall
move $a0,$v0      #spostamento di X in registro preservante
li $v0,5          #lettura secondo valore di ingresso
syscall
move $a1,$v0      #spostamento di Y in registro preservante
jal Funzione      #salto a funzione ricorsiva
move $a0,$v0      #recupero del valore di ritorno della funzione ricorsiva
li $v0,1          #stampa del risultato
syscall

li $v0, 10        #terminazione programma
syscall
```


Esercizio stack

Soluzione

Funzione:

```
blez $a0,casobase
subi $sp,$sp,16
sw $ra,0($sp)
sw $a0,4($sp)
sw $a1,8($sp)
sub $a0,$a0,5      #x=x-5
add $a1,$a1,3      #y=y+3
jal Funzione
lw $ra,0($sp)
lw $t0,4($sp)
lw $t1,8($sp)
add $sp,$sp,16
sub $v1,$t0,$t1    #(x-y)
mul $v0,$v0,$v1    #(x-y)*prodotti_precedenti
jr $ra
```

casobase:

```
li $v0,4
jr $ra
```

The background is a dark blue gradient with intricate white and yellow circuit board traces and components. The traces are thin lines that branch out and connect various components. The components are represented by small white and yellow shapes, including rectangles, circles, and lines, which are scattered across the background. The word "FINE" is centered in the middle of the image in a bold, yellow, sans-serif font.

FINE