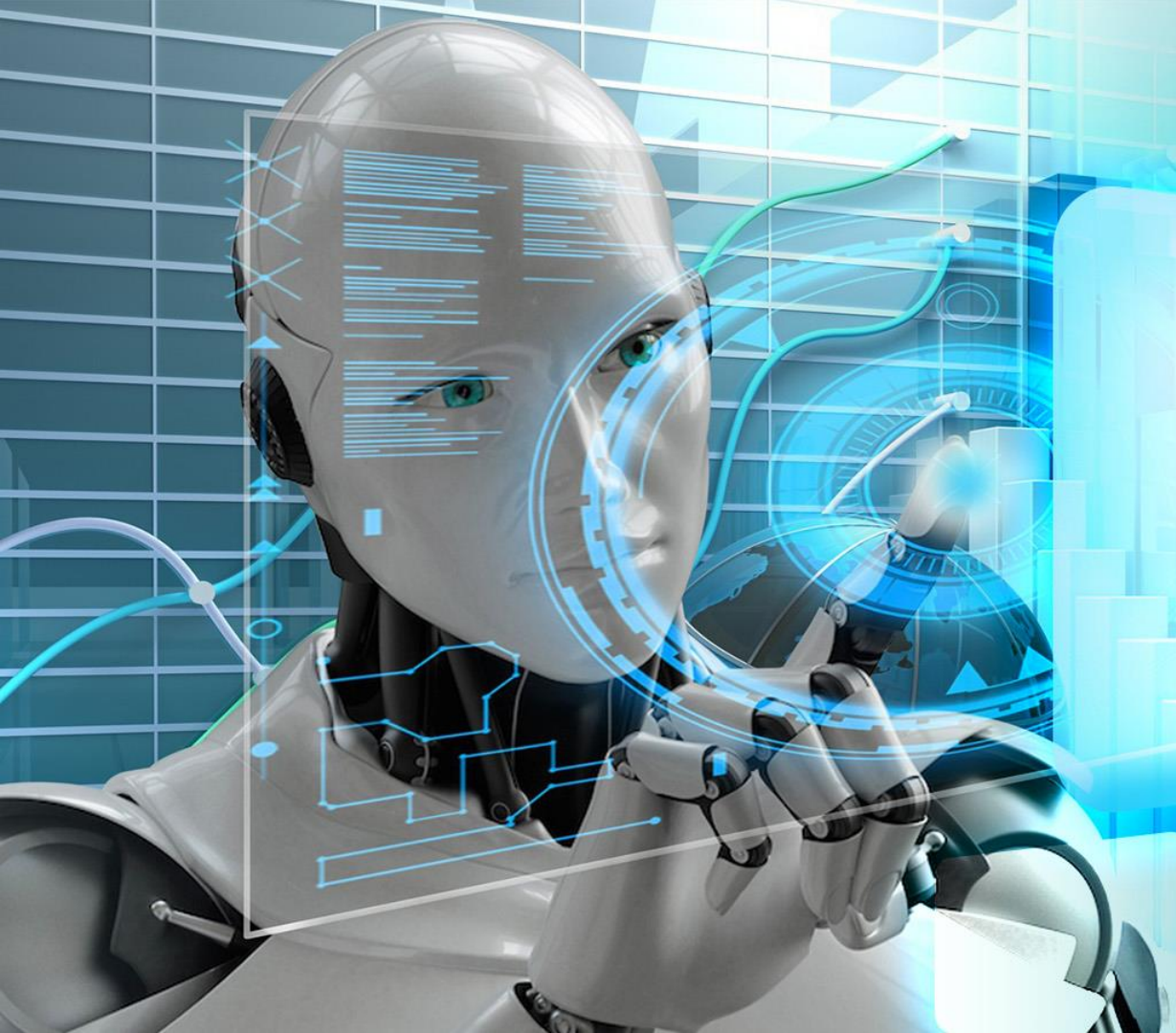




Architettura Elaboratori Elettronici ESERCITAZIONI STACK

Franco Liberati
liberati@di.uniroma1.it



STACK

Esercizio

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:

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

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



STACK 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

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

case_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)

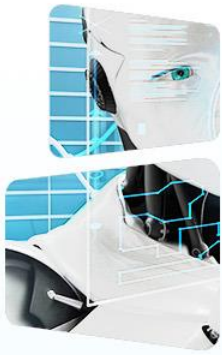
#POP dei valori precedentemente custoditi nello stack

#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



STACK

Esercizio

Scrivere il codice di una funzione ricorsiva che inverta le cifre di un numero intero N

```
int reverse (int num,int a) {  
    if (num==0) {return a;}  
    else {  
        return reverse(num/10,a*10+num%10);  
    }  
}
```



STACK main

main:

li \$v0,5

#lettura valore di ingresso

syscall

move \$a0,\$v0

#spostamento di X in registro preservante

li \$v0,0

move \$a1,0

#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



```
int reverse (int num,int a) {  
    if (num==0) {return a;}  
    else {  
        return reverse(num/10,a*10+num%10);  
    }  
}
```

STACK funzione

Funzione:

```
beqz $a0,caso_base  
rem $t1,$a0,10      #num%10  
mul $a1,$a1,10      #a*10  
add $a1,$a1,$t1     #a=a*10+num%10  
div $a0,$a0,10      #num=num/10  
sw $ra,($sp)        # PUSH ritorno  
sub $sp,$sp,4       #  
jal Funzione        # ricorsione  
add $sp,$sp,4       # POP  
lw $ra,($sp)        #  
move $v0,$v1        #salvo il valore del caso base in V0  
jr $ra
```

caso_base:

```
move $v1, $a1  
jr $ra
```



STACK

Esercizio

Scrivere il codice di una funzione ricorsiva `int f(int n)` che restituisce quante coppie di cifre uguali in posizioni adiacenti ci sono nel numero `n`, nel caso `n` sia negativo restituisce 0

Ad es: `f(551122)` restituisce 3, `f(5122)` restituisce 1, `f(9)` restituisce

`cd(n)`

{

 if (`n < 10`) return 0

 else{

`c1 = n % 10`

`n1 = n / 10`

`c2 = n1 % 10`

 if (`c1 == c2`) `cd(n1) + 1`

 else `cd(n1)`

 }

}



STACK 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



cd(n) {

if (n<10) return 0
else{

```
c1=n%10
n1=n/10
c2=n1%10
if (c1==c2) cd(n1)+1
else cd(n1)
}
```

STACK

Funzione ricorsiva

Funzione:

blt \$a0, 10, caso_base

rem \$t1,\$a0,10

div \$a0,\$a0,10

rem \$t2,\$a0,10

beq \$t1,\$t2, set_uno

li \$t9,0

j salta

set_uno:

li \$t9,1

salta:

sub \$sp,\$sp,8

sw \$t9,0(\$sp)

sw \$ra,4(sp)

jal Funzione

lw \$t0,0(\$sp)

lw \$ra,4(\$sp)

add \$sp,\$sp,8

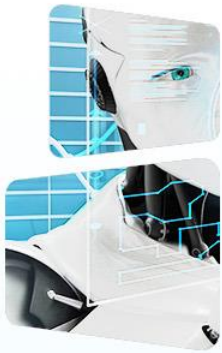
add \$v0,\$v0,\$t0

jr \$ra

caso_base:

li \$v0, 0

jr \$ra



STACK

Esercizio proposto

Si consideri la funzione f definita su interi

$$f(x,y) = 2 \cdot f(x-2,y-5)$$

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

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

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

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:

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

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

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



STACK
main

$$\begin{aligned}f(x,y) &= 2*f(x-2,y-5) \\ f(0,y) &= 1 \\ f(x,0) &= 2 \\ f(0,0) &= 3\end{aligned}$$

.text

.globl main

main:

li \$v0,5

syscall

move \$a0,\$v0

li \$v0,5

syscall

move \$a1,\$v0

jal Funzione

move \$a0,\$v0

li \$v0,1

syscall

li \$v0,10

syscall



$$\begin{aligned}f(x,y) &= 2*f(x-2,y-5) \\ f(0,y) &= 1 \\ f(x,0) &= 2 \\ f(0,0) &= 3\end{aligned}$$

STACK ricorsione

Funzione:

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

#confronto per stabilire il caso base
#

```
subu $sp, $sp, 4
```

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

```
sw $ra, 0($sp)
```

#

```
sub $a0,$a0,2
```

#aggiornamento X (x-2)

```
sub $a1,$a1,5
```

#aggiornamento Y (y-5)

```
jal Funzione
```

```
lw $ra, 0($sp)
```

#POP dei valori di RA precedentemente custoditi nello stack

```
addi $sp, $sp, 4
```

#

```
mul $v0,$v0,2
```

#costante da moltiplicare all'elemento selezionato

```
jr $ra
```

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



$$\begin{aligned}f(x,y) &= 2*f(x-2,y-5) \\ f(0,y) &= 1 \\ f(x,0) &= 2 \\ f(0,0) &= 3\end{aligned}$$

STACK

caso base

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



STACK

Esercizio proposto per casa

Si consideri la funzione f definita su interi

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

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

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:

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



$$f(x,y) = (x-y) \cdot f(x-5,y+3)$$
$$f(x,y) = 4 \text{ se } x \leq 0$$

STACK
main

.text

.globl main

main:

li \$v0,5

syscall

move \$a0,\$v0

li \$v0,5

syscall

move \$a1,\$v0

jal Funzione

move \$a0,\$v0

li \$v0,1

syscall

li \$v0,10

syscall



$$f(x,y) = (x-y) \cdot f(x-5,y+3)$$
$$f(x,y) = 4 \text{ se } x \leq 0$$

STACK ricorsione e caso base

Funzione:

```
blez $a0,casobase
subi $sp,$sp,12
sw $ra,0($sp)
sw $a0,4($sp)
sw $a1,8($sp)
sub $a0,$a0,5
add $a1,$a1,3
jal Funzione
lw $ra,0($sp)
lw $t0,4($sp)
lw $t1,8($sp)
add $sp,$sp,12
sub $v1,$t0,$t1 # (x-y)
mul $v0,$v0,$v1
jr $ra
```

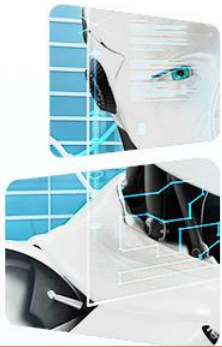
#x=x-5

#y=y+3

#(x-y)*prodotti_precedenti

casobase:

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



STACK

Esercizio proposto per casa

Progettare e codificare un programma in assembly MIPS/MARS che calcola il quoziente della divisione tra interi in modo ricorsivo, cioè

```
int div(int x, int y){  
    if (x>=y) return 1+div(x-y,y);  
    else return 0;  
}
```



```
int div(int x, int y){  
    if (x>=y) return 1+div(x-y,y);  
    else return 0;  
}
```

STACK
main

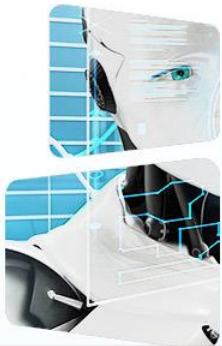
QUOZ:

```
.text  
.globl main  
main:  
    lw $a0,divid  
    lw $a1,divis  
    jal QUOZ  
    move $a0,$v0  
  
    li $v0,1  
    syscall  
  
li $v0,10  
syscall
```

```
blt $a0,$a1,casobase    # analisi casobase  
sub $sp,$sp,12          # PUSH salvataggio n e indirizzo di ritorno  
sw $ra,0($sp)           #  
sw $a0,4($sp)           #  
sw $a1,8($sp)           #  
sub $a0,$a0,a1          #  
jal QUOZ  
lw $ra,0($sp)           #  
lw $a0,4($sp)           #  
lw $a1,8($sp)           #  
add $sp,$sp,12          # POP  
add $v0,$v0,1           # aggiornamento somma parziale  
jr $ra
```

casobase:

```
li $v0,0                # gestione caso base  
jr $ra
```

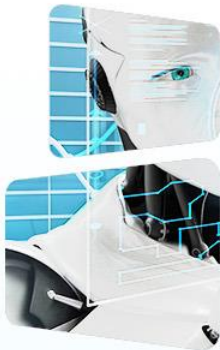


STACK

Esercizio proposto per casa

Progettare e codificare un programma in assembly MIPS/MARS che determina se un numero è primo in modo ricorsivo, cioè (1 NON PRIMO, 0 PRIMO)

```
bool isp(int n, int i){  
    if (n%i)!=0 &&(n>i) isp(i+1,n)  
    else return 1  
}
```



.text

.globl main

main:

```
lw $a0,n
li $a1,2
li $v0,0
jal ISPRIM
move $a0,$v0
li $v0,1
syscall
```

```
li $v0,10
syscall
```

ISPRIM:

casobase:

fine:

```
bool isp(int n, int i){
    if (n%i)!=0 isp(i+1,n)
    else return 1
}
```

```
ble $a0,$a1,fine
rem $t0,$a0,$a1
beqz $t0, casobase          # analisi casobase
sub $sp,$sp,12              # PUSH salvataggio n e indirizzo di ritorno
sw $ra,0($sp)               #
sw $a0,4($sp)               #
sw $a1,8($sp)               #
add $a1,$a1,1               #
jal ISPRIM
lw $ra,0($sp)               #
lw $a0,4($sp)               #
lw $a1,8($sp)               #
add $sp,$sp,12              # POP
jr $ra

casobase:
li $v0,1                    # gestione caso base

fine:
jr $ra
```

STACK
main



STACK

Esercizio

Su una isola c'è una coppia di giovani conigli, i coniglietti il primo anno sono troppo giovani e non fanno figli mentre una coppia di conigli ogni anno a partire dal secondo fa un'altra coppia di conigli

Dopo N anni quante coppie sono presenti sull'isola?

❖ Cosa succede:

Conigli(0) = 1

la prima coppia è stata messa sull'isola

Conigli(1) = 1

al primo anno la prima coppia è ancora giovane

Conigli(2) = 1 + 1

al secondo anno la coppia iniziale genera una seconda coppia

Conigli(3) = 2 + 1

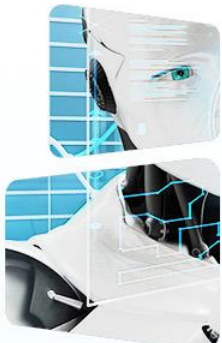
la seconda coppia non è matura, la prima coppia genera ancora

Conigli(4) = 3 + 2

tutte le coppie che hanno almeno 2 anni generano altre coppie

Conigli(5) = 5 + 3

....



STACK

Esercizio

Isola dei conigli

❖ Cosa succede:

$$\text{Conigli}(0) = 1$$

$$\text{Conigli}(1) = 1$$

$$\text{Conigli}(2) = 1 + 1$$

$$\text{Conigli}(3) = 2 + 1$$

$$\text{Conigli}(4) = 3 + 2$$

$$\text{Conigli}(5) = 5 + 3$$

la prima coppia è stata messa sull'isola

al primo anno la prima coppia è ancora giovane

al secondo anno la coppia iniziale genera una seconda coppia

la seconda coppia non è matura, la prima coppia genera ancora

tutte le coppie che hanno almeno 2 anni generano altre coppie

....

Conigli

$$f(n) = f(n-1) + f(n-2)$$

$$f(1) = 1$$

$$f(0) = 1$$



$$f(n) = f(n-1) + f(n-2)$$
$$f(1) = 1$$
$$f(0) = 1$$

STACK
main

```
text
.globl main
main:
    li $v0,5
    syscall
    move $a0,$v0

    jal CONIGLI

    move $a0,$v0
    li $v0,1
    syscall

    li $v0,10
    syscall
```

CONIGLI:

```
ble $a0,1,casobase          # analisi casobase
sub $sp,$sp,18 # PUSH salvataggio n e indirizzo di ritorno (prima volta $RAmain)
sw $ra,0($sp)                # e poi n-1 e salvataggio a RA1
sw $a0,4($sp)
sub $a0,$a0,1                # n-1
jal CONIGLI
sw $v0,8($sp)                # salvataggio somma parziale
lw $a0,4($sp)                # prelievo n da analizzare
sub $a0,$a0,2                # n-2
jal CONIGLI
lw $t0,8($sp)                # prelievo somma parziale
add $v0,$v0,$t0              # aggiornamento somma parziale
lw $ra,0($sp)                # recupero RA
add $sp,$sp,8 # POP
jr $ra
```

1

2

casobase:

```
li $v0,1                    # gestione caso base
jr $ra
```



text
.globl main
main:

li \$v0,0

li \$v0,5
syscall
move \$a0,\$v0

jal FIBONACCI

move \$a0,\$v0
li \$v0,1
syscall

li \$v0,10
syscall

FIBONACCI:

$$f(n) = f(n-1) + f(n-2)$$
$$f(1) = 1$$
$$f(0) = 1$$

ble \$a0,1,casobase
sub \$sp,\$sp,12
sw \$ra,0(\$sp)
sw \$a0,4(\$sp)
sub \$a0,\$a0,1
jal FIBONACCI
sw \$v0,8(\$sp)
lw \$a0,4(\$sp) **a**
sub \$a0,\$a0,2
jal FIBONACCI
lw \$t0,8(\$sp) **b**
add \$v0,\$v0,\$t0
lw \$ra,0(\$sp)
add \$sp,\$sp,12
jr \$ra

casobase:

li \$v0, 1
jr \$ra

-
2
R a
-
3
R a
-
4
Rmain

1
2
R a
-
3
R a
-
4
Rmain

2
3
R a
-
4
Rmain

3
4
Rmain

-
2
R b
3
4
Rmain

4
4
Rmain

FINE

