Architettura Elaboratori Elettronici

ESERCITAZIONI

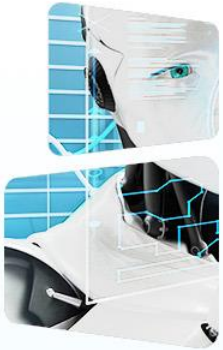
COPROCESSORE MATEMATICO

Franco Liberati

liberati@di.uniroma1.it

Argomenti

- Il coprocessore matematico del MARS
- Istruzioni



Il Coprocessore Matematico nel MARS

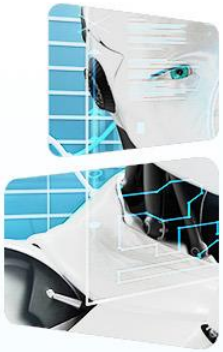
- Il MIPS è dotato anche di un **coprocessore matematico** che svolge operazioni in virgola mobile a singola precisione (float) e doppia (double) precisione

- Il MIPS è dotato di 32 registri per i calcoli in virgola mobile: **$f0** - **$f31**

- Di solito le istruzioni hanno lo stesso menmonico del corrispondente tra interi seguito da **.s** per i calcoli in singola precisione o da **.d** per quelli in doppia precisione
- La definizione di un valore reale in memoria si ottiene con la direttiva

  .float

  .double



Il Coprocessore Matematico nel MARS

Coprocessor 1
REGISTRI

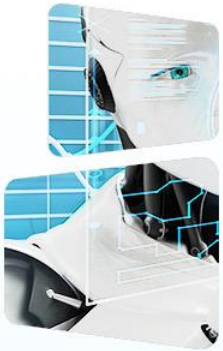
$f0
$f1
...
$f31

CU

ALU

ESEMPIO
.data
pippo: .**float** 3.589

pluto: .**double** 457435343523.5646489

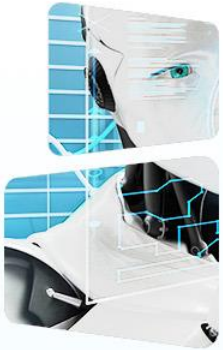


Il Coprocessore Matematico nel MARS

(Esempio di somma tra due numeri reali definiti in memoria)

```
.text
.globl main

main:
        lwc1 $f1,r1          #spostamento del valore nel registro del coprocessore $f1
        lwc1 $f2,r2          #spostamento del valore nel registro del coprocessore $f2
        add.s $f0,$f1,$f2    #somma tra $f1 e $f0 e restituzione del risultato in $f0

        li $v0,10
        syscall

.data
r1: .float 3.5
r2: .float -23.7
```

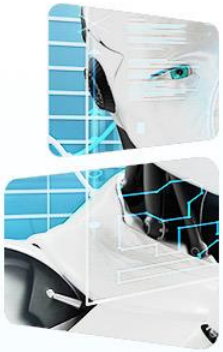
ISTRUZIONI PER NUMERI IN VIRGOLA MOBILE



Istruzioni per numeri in Virgola Mobile

(spostamento dalla e alla memoria)

lwc1 rd,<imm>|<label>	Preleva un valore reale <imm> o il contenuto di una etichetta <label> definita in memoria con un operando reale in un registro floating point rd=<imm>
swc1 rs,<label>	Archivia il valore reale sito in un registro floating point in memoria MEM=rs
l.d rd,<imm>|<label>	Preleva un valore reale <imm> o il contenuto di una variabile <label> definita in memoria in due registri floating point (rd e rd+1) doppia precisione rd=<imm>
l.s rd,<imm>|<label>	Preleva un valore reale <imm> o il contenuto di una variabile <label> definita in memoria in un registro floating point singola precisione rd=<imm
s.d rs, <label>	Archivia il valore reale di due registro floating point (rs e rs+1) doppia precisione in memoria
s.s rs, <label>	Archivia il valore reale di un registro floating point singola precisione in memoria

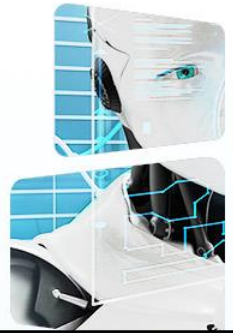


Istruzioni per numeri in Virgola Mobile

(spostamento dalla e alla memoria)

```
l.d $f2, reale64bit     # Nei registri $f2 e $f3 è presente il valore in doppia precisione contenuto
                        #nella variabile reale64bit
```

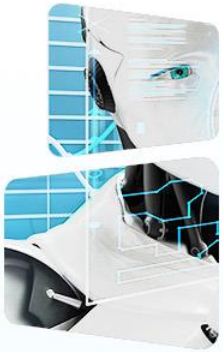
```
s.d $f4, reale64bit     # In otto locazioni contigue di memoria alla posizione indicata
                        #dall'etichetta reale64bit è presente il valore contenuto in $f4 e $f5
```



Istruzioni per numeri in Virgola Mobile

(spostamento tra registri)

mov.s rd,rs	**Sposta il contenuto tra registri floating point (singola precisione) rd=rs**
mov.d rd,rs	**Sposta il contenuto tra registri floating point (doppia precisione) rd=rs**
mfcz rdCU,rs **mfcz.d rdCU,rs**	**Sposta il contento del registro rs del coprocessore *z* nel registro destinazione rd della CU** **Se si sposta un double il contenuto al termine dell'operazione si trova in rd e rd+1** ***(NON SI CONVERTE IL NUMERO)***
ESEMPIO MARS **mfc1 $t0,$f0** **#Sposta il valore float da $f0 a $t0** **mfc1.d $t0,$f0** **#Sposta il valore double di $f0 e $f1 in $t0 e $t1**	
mtcz rsCU, rd **Mtcz.d rsCU,rd**	**Sposta il contento del registro rs della CU nel registro destinazione rd del coprocessore *z*** ***(NON SI CONVERTE IL NUMERO)***
ESEMPIO MARS (coprocessore 1) **mtc1 $t0,$f0** **#Sposta il valore da $t0 a $f0**	



Istruzioni per numeri in Virgola Mobile

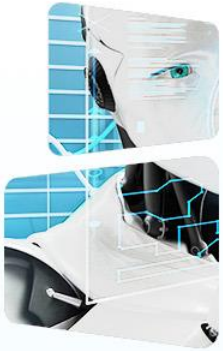
(conversione dei valori)

cvt.d.s rd,rs	**Converte da singola precisione a doppia precisione rd=(double)rs**
cvt.d.w rd,rs	**Converte da intero a 64bit double**
cvt.s.d rd,rs	**Converte da doppia precisione a singola precisione rd=(float)rs**
cvt.s.w rd,rs	**Converte da intero a 32bit float**
cvt.w.d rd,rs	**Converte da 64bit double a intero**

Istruzioni per numeri in Virgola Mobile

Istruzioni Aritmetiche

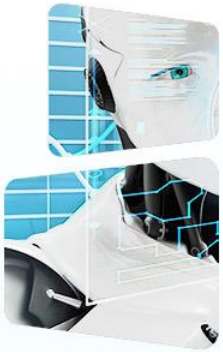
abs.s regd,regs	Valore assoluto di regs in regd per numero float
abs.d regd,regs	Valore assoluto di regs(e regs+1) in regd (e regd+1) per numero double
add.s regd,reg1,reg2	Somma per numeri float: regd=reg1+reg2
add.d regd,reg1,reg2	Somma per numeri double: regd(e regd+1)=reg1(e reg1+1)-reg2(e reg2+1)
sub.s regd,reg1,reg2	Sottrazione per numeri float: regd=reg1+reg2
sub.d regd,reg1,reg2	Sottrazione per numeri double: regd(e regd+1)=reg1(e reg1+1)-reg2(e reg2+1)
mul.s regd,reg1,reg2	Moltiplicazione tra numeri float: regd=reg1*reg2
mul.d regd,reg1,reg2	Moltiplicazione tra numeri double: regd (e regd+1)=reg1(e reg2+1)*reg2(e reg2+1)
div.s regd,reg1,reg2	Divisione tra numeri float: regd=reg1/reg2
div.d regd,reg1,reg2	Divisione tra numeri double: regd=reg1 (e reg1+1)/reg2(e reg2+1)
sqrt.s regd,regs	Radice quadrata di numero float in regs (risultato in regd)
sqrt.d regd,regs	Radice quadrata di numero double in regs e regs+1(risultato in regd)



Istruzioni per numeri in Virgola Mobile

Istruzioni Aritmetiche (approssimazione)

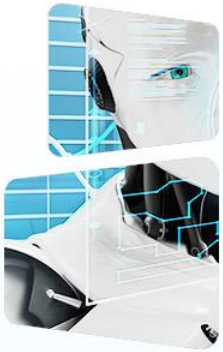
ceil.w.s regd,regs	Parte intera superiore del valore float regs nel registro regd
ceil.w.d regd,regs	Parte intera superiore del valore double regs nel registro regd
floor.w.s regd,regs	Parte intera del valore float regs nel registro regd
floor.w.d regd,regs	Parte intera del valore double regs nel registro regd
trunc.w.s regd,regs	Troncamento del valore float regs nel registro regd
trunc.w.d regd,regs	Troncamento del valore double regs nel registro regd



Istruzioni per numeri in Virgola Mobile

ESEMPIO I

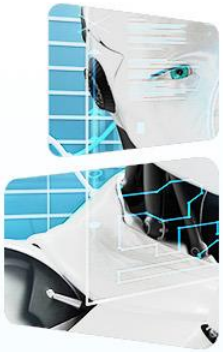
Si scriva un programma in linguaggio assembly che definiti due numeri reali in memoria r1 e r2 riporta la media (fra reali) in $f2



Istruzioni per numeri in Virgola Mobile

ESEMPIO I

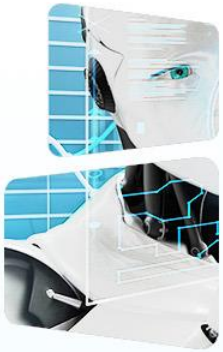
```
.text
.globl main

main:
        lwc1 $f0,r1          #carico r1
        lwc1 $f1,r2          #carico r2
        lwc1 $f3,due         #carico il valore 2.0
        add.s $f2,$f0,$f1    #m=x+y
        div.s $f2,$f2,$f3    #media
        li $v0,10
        syscall

.data
r1 : .float 5.0
r2: .float  3.0
due: .float 2.0
```



Istruzioni per numeri in Virgola Mobile

ESEMPIO II

Si scriva un programma in linguaggio assembly che definiti due numeri interi in memoria val1 e val2 riporta la media in memoria



Istruzioni per numeri in Virgola Mobile

ESEMPIO II

```
.text
.globl main

main:
        lw $t0,val1                 #carico val1
        lw $t1,val2                 #carico val2
        add $t0,$t1,$t0             #sommo i valori
        mtc1 $t0,$f0                #sposto il valore in $f0 (senza convertirlo)
        cvt.s.w $f0,$f0             #converto il valore in standard IEEE 754
        lwc1 $f3,due                #carico il valore 2.0
        div.s $f2,$f0,$f3           #media
        swc1 $f2, media
        li $v0,10
        syscall

.data
val1: .word 3455
val2: .word  4562
due: .float 2.0
media:.float 0.0
```

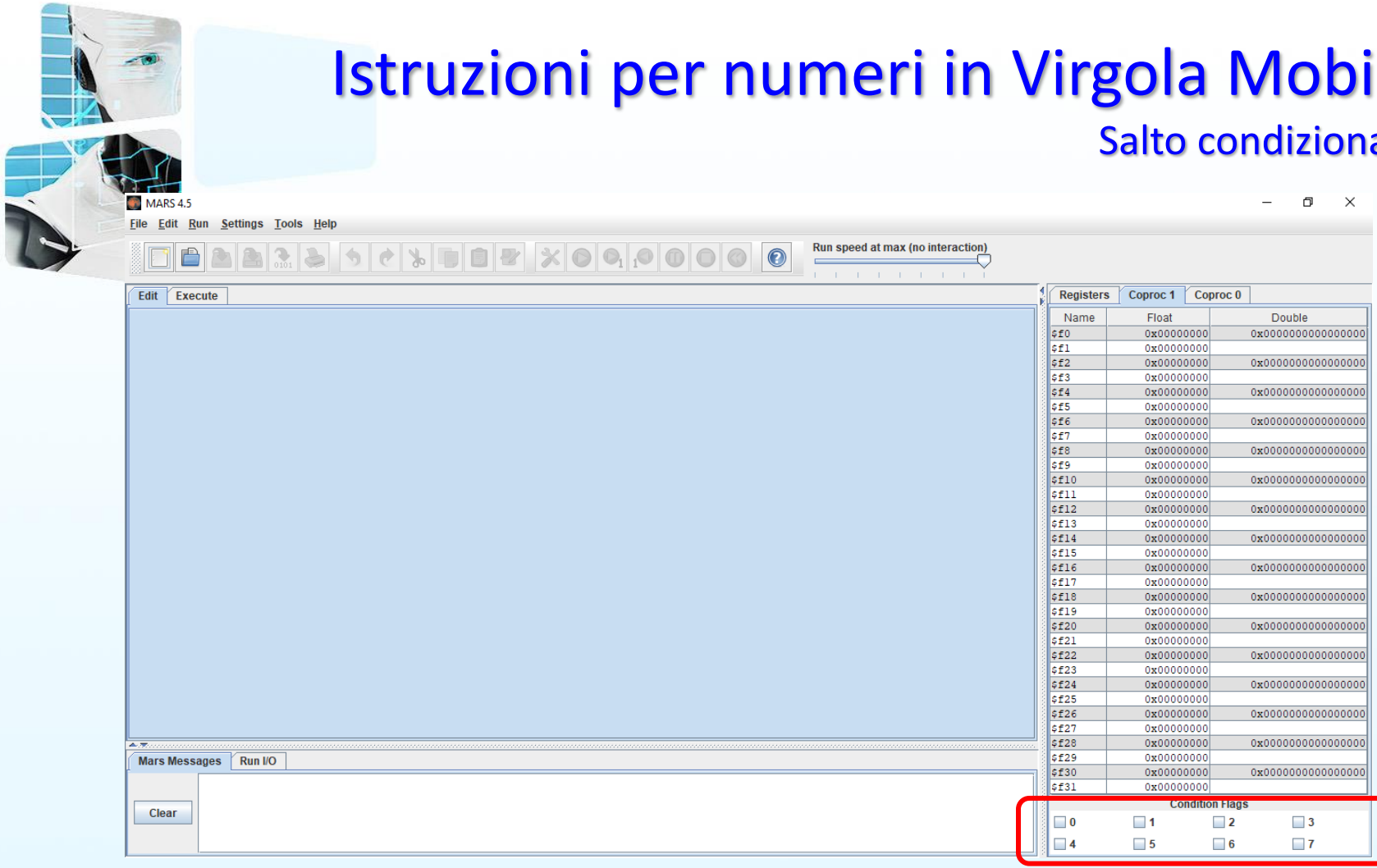
Istruzioni per numeri in Virgola Mobile

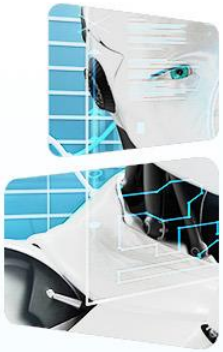
Salto condizionato

c.eq.s imm,reg1,reg2	Se reg1=reg2 (float) allora setta a 1/TRUE/Spunta il flag specificato da <imm>
c.eq.d imm,reg1,reg2	Se reg1=reg2 (double) allora setta a 1/TRUE/Spunta il flag specificato da <imm>
c.le.s imm,reg1,reg2	Se reg1<=reg2 (float) allora setta a 1/TRUE/Spunta il flag specificato da <imm>
c.le.d imm,reg1,reg2	Se reg1<=reg2 (double) allora setta a 1/TRUE/Spunta il flag specificato da <imm>
c.lt.s imm,reg1,reg2	Se reg1<reg2 (float) allora setta a 1/TRUE/Spunta il flag specificato da <imm>
c.lt.d imm,reg1,reg2	Se reg1<reg2 (double) allora setta a 1/TRUE/Spunta il flag specificato da <imm>
bc1f <imm>, label	Salta a label se il *codice di condizione* indicato da <imm> è settato a 0 (significato logico 0/FALSE/NonSpuntato)
bc1t <imm>, label	Salta a label se il *codice di condizione* indicato da <imm> è settato a 1 (significato logico 1/TRUE/Spuntato)

Istruzioni per numeri in Virgola Mobile

Salto condizionato

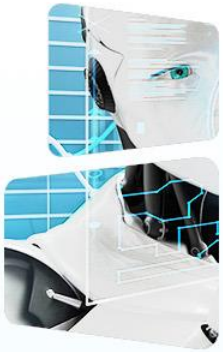




Istruzioni per numeri in Virgola Mobile

ESEMPIO III

Si scriva un programma in linguaggio assembly che letti quattro numeri interi da tastiera effettua la media dei primi due numeri e la media dei secondi due numeri e pone in $t9 il valore 1 se la prima media è maggiore della seconda altrimento $t9 è impostato a 0



Istruzioni per numeri in Virgola Mobile

ESEMPIO III

```
li $v0,5          #Leggo il primo valore
syscall
move $t0,$v0
li $v0,5          #Leggo il secondo valore
syscall
move $t1,$v0
add $t0,$t0,$t1#sommo i valori
mtc1 $t0,$f0    #sposto il valore in $f0 (senza convertirlo)
cvt.s.w $f0,$f0 #converto il valore in standard IEEE 754
lwc1 $f3,due    #carico il valore 2.0
div.s $f0,$f0,$f3 #media in $f0
```

```
li $v0,5          #Leggo il terzo valore
syscall
move $t2,$v0
li $v0,5          #Leggo il quarto valore
syscall
move $t3,$v0
add $t2,$t2,$t3 #sommo i valori
mtc1 $t2,$f1    #sposto il valore in $f0 (senza convertirlo)
cvt.s.w $f1,$f1 #converto il valore in standard IEEE 754
lwc1 $f3,due    #carico il valore 2.0
div.s $f1,$f1,$f3 #media in $f1
```

```
#metto a 1/TRUE/SPUNTA il flag se la prima media è minore della prima
                c.lt.s 3,$f0,$f1
                li $t9,1
#salto se il flag 3 è settato a 0/FALSE/NOSPUNTA (non è flaggato) cioè $f0>$f1 ovvero la prima media e maggiore della seconda
                bc1f 3, fine
                li $t9,0
fine:
                li $v0,10
                syscall
```

```
.data
due: .float 2.0
```

FINE