ISTRUZIONI D-RISC

lunedì 23 ottobre 2017 08:44

→ ARITMETICO LOGICHE

→ MEMORIA

→ SALTO { CONDIZIONALI / INCONDIZIONALI
"SPECIALI"

"fanno di conto"

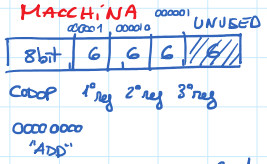
trasferimento registri <-> memoria

while (true) {
prelievo dell'ISTR M[IC]
decodifica
esecuzione
fatti interni
}

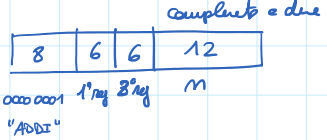
ISTRUZIONE D-RISC

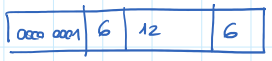
SINTASSI "ASSEMBLER" | SEMANTICA | FORMATO LING. MACCHINA

ADD R_i, R_j, R_K — $R[i]+R[j] \to R[k]$

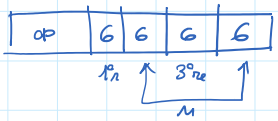


(ADDI)
ADD $R_i, \#n, R_K$ — $R[i]+n \to R[K]$

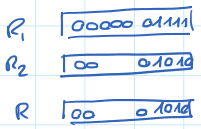




SUB
- SUB R_i, R_j, R_K — $R[i]-R[j] \to R[K]$ — 0000 0010
- SUBI $R_i, \#n, R_K$ — $R[i]-n \to R[k]$ — 0000 0011



(MUL, DIV)



AND — AND R_1, R_2, R_3

OR

NOT

SHR SHL

SHR R_i, R_K / $R_i, \#n$ — Shift di R_K pos verso R

SHR $R_1, \#n, R_K$ — SHR (R_1 x n pos) → R_K

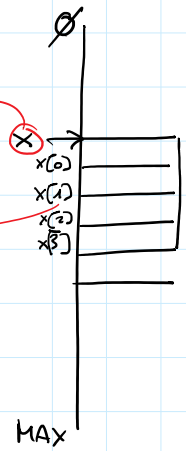
Istruzioni Memoria

lunedì 23 ottobre 2017 09:47

x[N]
N=4

(LD)
LOAD $R_i, R_K R_j$ $M[R[i]+R[k]] \rightarrow R[j]$

STORE
(ST)

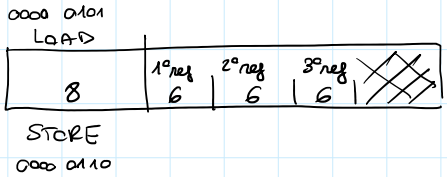


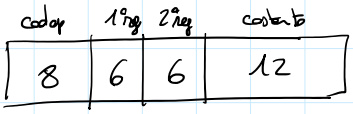
LOAD $R_{base}, R_{indice}, R_{dest}$

LOAD $R_{base}, \#n, R_{dest}$

$M[R(base)+n] \rightarrow R[dest]$

STORE $R_i, \begin{matrix} R_K \\ \#n \end{matrix}, R_j$ $M[R[i]+\begin{matrix} R[k] \\ n \end{matrix}] \leftarrow R[j]$





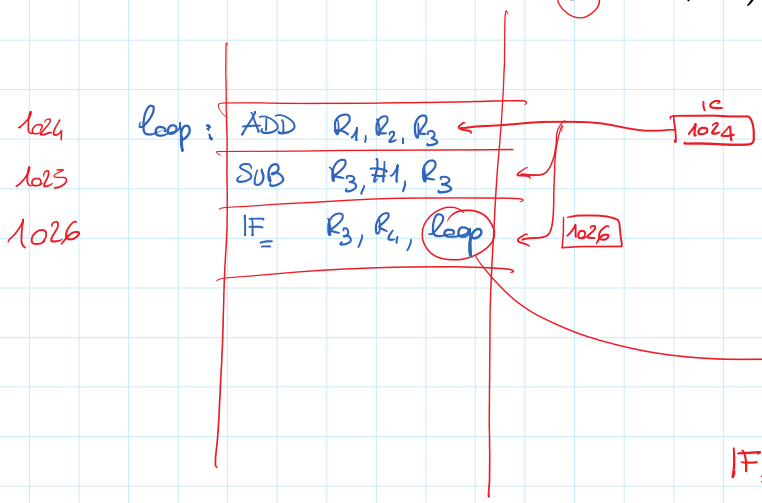
istruzioni di salto

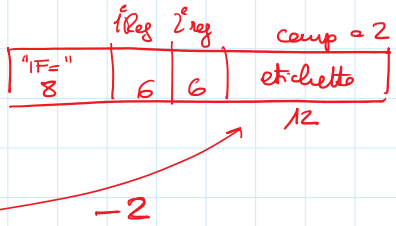
lunedì 23 ottobre 2017 09:57

salto condizionato $IF_{=}\ R_i, R_k$, etichetta

$R_3 == R_4 \Rightarrow IC + etichetta \rightarrow IC$

$R_3 != R_4 \Rightarrow IC + 1 \rightarrow IC$

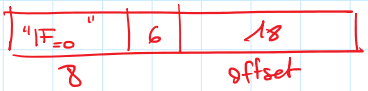




−2

$IF_{=}$ $IF_{<}$ $IF_{>}$ $IF_{!=}$

$IF_{=0}$ $IF_{<0}$ $IF_{>0}$



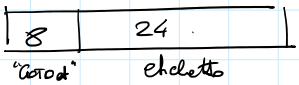
SALTI INCONDIZIONATI

GOTO

CALL

GOTO Reg i $IC \leftarrow REG[i]$

GOTO etichetta $IC \leftarrow IC + etichetta$



-2^{23}

$+2^{23} - 1$

IC
1024
→ CALL R_f, R_{ret}

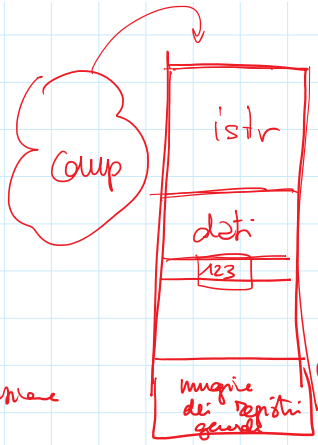
2096

↑ "link register"

2096 $IC \leftarrow R[f]$; 1025 $R[ret] \leftarrow IC + 1$

Compilazione
lunedì 23 ottobre 2017 10:27

```
int x = 123;
int y[N];
for(int i = 0; i < N; i++)
    y[i] = i;
```

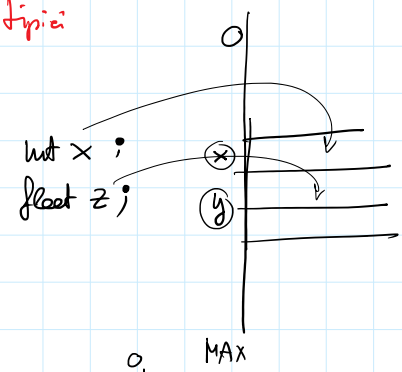


gcc -s m.c

Rappresentazione dei dati

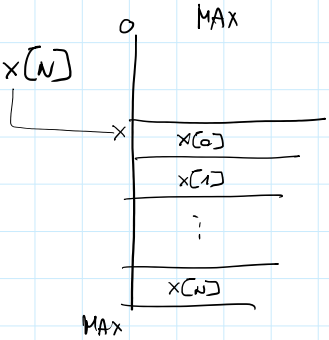
Compilazione degli statement tipici

→ scalari



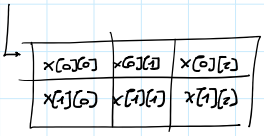
→ vettori

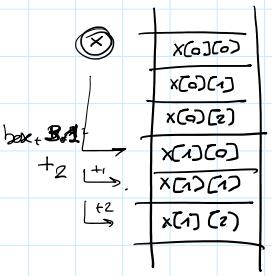
int x[N]



array x[N][M] (2 3)

"row first"



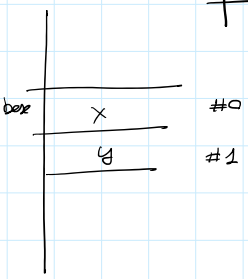


x_{ij} base + M*i + j

1,2

→ struct

```
struct { int x;
         float y; }
```

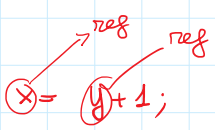


compilazione degli statement tipici

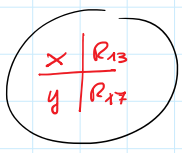
→ assegnamento → condizionali { if then / if then else } → cicli { for / while } → procedure/funzioni

→ scalari

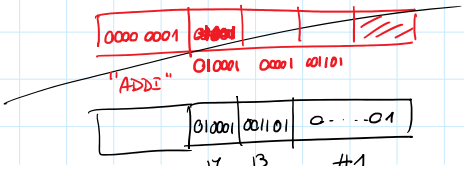
vettori



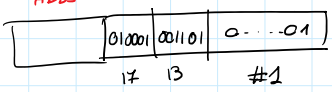
ADD Ry, #1, Rx



ADD R17, #1, R13



"ADDI"




```
X[i] = a[i] + b[i] ;
```

∀ vettore abbiamo un registro base (universalmente dal compilatore)

R_{basea} R_{baseb} R_{basex}

```
LOAD   Rbasea, Ri, Rai
LOAD   Rbaseb, Ri, Rbi
ADD    Rai, Rbi, Rtemp
STORE  Rbasex, Ri, Rtemp
```

condizionale (compilazione)

lunedì 23 ottobre 2017 10:50

if(e) then Statement

IF compilazione della condizione !e , "cont"

{ compilazione di Statement }

cont : ....

if (x=0) then {x = x+1;}

IF≠0 Rx, "cont"
ADD Rx, #1, Rx
cont : ---

IF=0 Rx, "then"

then : { compilazione di x=x+1; }
GOTO cont:

cont : ....

if (e) then {S1} else {S2}

e → IFe , then
compilazione {S2}
GOTO fine
then : compilazione {S1}
fine : ---

¬e

if (x=0) then {x = x+1;} (S1) else {x = x-1;} (S2)

IF=0 Rx, "then"
SUB Rx, #1, Rx
GOTO "fine"
"then" : ADD Rx, #1, Rx
"fine" : — —

Compilazione di cicli deterministici (For)

lunedì 23 ottobre 2017 10:58

```
for(int i=0; i<N; i++) { x[i] = i; }
```

inizializzo la var di iterazione

loop: {compilazione del corpo del for}

aggiornamento della var di iterazione

IF$_{\leq}$, loop

cont: ----

```
ADD R0, R0, Ri
loop: STORE Rbasex, Ri, Ri
      ADD Ri, #1, Ri
      IF< Ri, RN, loop
cont: ---
```