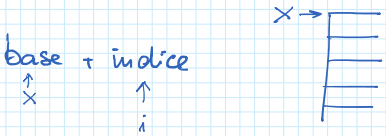
OPERANDI:

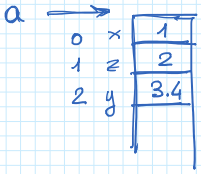
martedì 25 ottobre 2016 11:11

numero

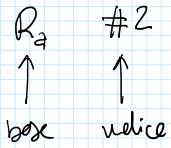
immediati

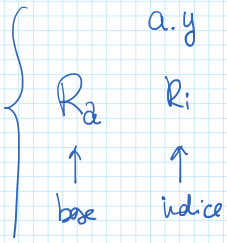



```
struct { int x;
         int z;
         float y; } a;
```



a.x (a→x)
a.z
a.y





2 → Ri

D-RISC

martedì 25 ottobre 2016 11:19

D → Didattico

RISC → Reduced Instruction Set Computer ←

↕ ↕

CISC Complex " " "

Registri R_0 R_1 ..... R_{max} general purpose

64 da 32 bit

$R_\emptyset$ ha sempre il valore $\emptyset$

R_{60} R_{61} R_{62}

Istruzioni che accedono alla memoria

solo due istruzioni che accedono la memoria

R[i] → MEM[ind] STORE

MEM[ind] → R[i] LOAD

unico indirizzamento (registro base + indice)

Istruzioni
- operative
  - operazioni aritmetiche logiche fra registri
  - R_i (op) R_j → R_K
- load/store
  - caricano dati da e per la memoria
- salto
  - salto condizionato
  - incondizionato

memoria centrale indirizzabile "alla parola"

ISTRUZIONI
- → FORMATO TXT (sintassi)
- → semantica
- → FORMATO in MEMORIA

ISTRUZIONI OPERATIVE (ARITMETICO/LOGICHE)

martedì 25 ottobre 2016 11:42

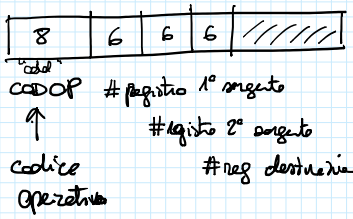
addizione

"sintassi": ADD oper1, oper2, oper3 → ADD R_5, R_{27}, R_3

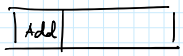
oper$_i$ sarà il numero di un registro

"semantica": REG[5] + REG[27] → REG[3]

"formato in memoria"



32 bit

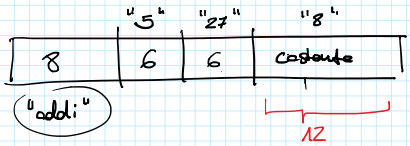


"sintassi" ADDI oper1, oper2, oper3

oper$_1$ e oper$_3$ sono registri
oper$_2$ è un op. immediato (costante)

ADDI R_5, #8, R_{27}

semantica REG[5] + 8 → REG[27]



ADD SUB MUL DIV SHR SHL

SHR R_1, R_2, R_3

REG[1] shiftato a destra di REG[2] posizioni e salvo il risultato in REG[3]

R_1 = 000110...
R_2 = 111101...
R = 000100

AND OR NOT bit a bit

AND R_1, R_2, R_3

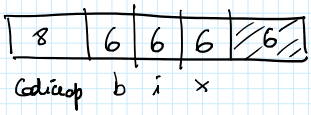
LOAD / STORE

martedì 25 ottobre 2016 11:55

sintassi

LOAD R_b, R_i, R_x — $MEM[REG[b] + REG[i]] \rightarrow REG[x]$

STORE R_b, R_i, R_x — $REG[x] \rightarrow MEM[REG[b] + REG[i]]$



EXCHANGE R_b, R_i, R_x — $MEM[REG[b] + REG[i]] \leftrightarrow REG[x]$

↳ (TEST & SET)

SALTI

martedì 25 ottobre 2016 12:15

CONDIZIONALI

sintassi: $IF_>$ R_i, R_j, etichetta (offset rispetto a IC)

semantica: se REG[i] > REG[j] allora IC + offset → IC altrimenti IC+1 → IC

formato:

8	6	6	12
Codop	i	j	offset

offset ← complemento a 2 $(-2^{11} .. 2^{11}-1)$

IC →

8			
$IF_>$	4	12	+3
SUB	- - -		
STORE	- - -		
ADD			

$IF_>$ R_4, R_{12}, continua
SUB
STORE
continua: ADD - - -

offset (−4) loop: ——
−3 ——
−2 ——
−1 ——
$IF_<$ R_i, R_N, loop

$IF_{>\emptyset}$ R_i, offset

se REG[i] > ∅ allora... ...

8	6	18
8	i	offset

Codop

Incondizionali

GOTO
- etichetta (offset)
- R_i

GOTO fine
IC ← IC + fine

GOTO R_{18}
IC ← REG[18]

chiamata di procedura

CALL R_F, R_{ret}

R_{ret} ← indirizzo di ritorno
R_F ← indirizzo della procedura/funzione da chiamare

IC ← REG[f], IC $\xrightarrow{+1}$ REG[ret]
sono "contemporanee"

ritorno di procedura

GOTO R

COMPILAZIONE

martedì 25 ottobre 2016 12:15

assegnamento

① `int z, x;`

`x = 0;`
`x = z;`
`x = 123;`

② `int y[16];`

`y[8] = 0;`
`x = y[8];`

① R_x registro che contiene x

ADD R_0, R_0, R_x x=0;
ADD R_z, R_0, R_x x=z
ADD R_0, #123, R_x x=123

② R_{basex} indirizzo di partenza del vettore in memoria

STORE R_{basex}, #8, R_0 y[8]=0;
LOAD R_{base}, #8, R_x x=y[8];

if-then-(else) for( ) while procedure

if(E) { Body }

if(x==0) { y = x+1; }
E Body

IF not(E), "cont"
Compilazione del Body
cont: — resto del programma

Schema di compilazione di un if-then

IF≠0 R_x, cont
ADD R_x, #1, R_y
cont: .....

"IF≠0"	x	+2	
"add:"	x	y	1

if(E) { Body-then } else { Body-else }

non E
E = True

IF E, then
compilazione Body-else
GOTO cont
then: compilazione Body-then
cont: .....

if(x==0) { y = x+1; }
else { y = x-1; }

IF=0 R_x, then
else: SUB R_x, #1, R_y
GOTO cont
then: ADD R_x, #1, R_y
cont: .....

while (x < N) { y = y + x; x = x*2; }

while (E) { Body }

while: IF not(E), cont
compilazione del Body

while: IF≥ R_x, R_N, cont
ADD R_y, R_x, R_y
MUL R_x, #2, R_x
} compilazione Body

while : IF not(e), cout

compilazione del Body

GOTO while

Cout : .....

MUL Ry, -x, -y
MUL Rx, #2, Rx
GOTO while

} compilazione Body

Cout :

for(i = 0 ; i < N ; i++) { Body }

compilazione init i

for : compilazione del Body

compilazione "INCR"

se Cond, for

Cout : .....

X[i] = 0;

ADD R0, R0, Ri
for : STORE Rbasex, Ri, R0
ADD Ri, #1, Ri
IF< Ri, RN, for

Cout : ....

x[i]=0
i++

$X_0\ X_1 \dots X_{n-1}$ $\quad y_0 \dots y_{n-1}$

$IP = \sum_{i=0}^{n-1} x_i y_i$

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

ADD R0, R0, Rsum
ADD R0, R0, Ri
for : LOAD Rbasex, Ri, Rxi
LOAD Rbasey, Ri, Ryi
MUL Rxi, Ryi, Rmul
ADD Rsum, Rmul, Rsum
ADD Ri, #1, Ri
IF< Ri, RN, for

Cout : ...