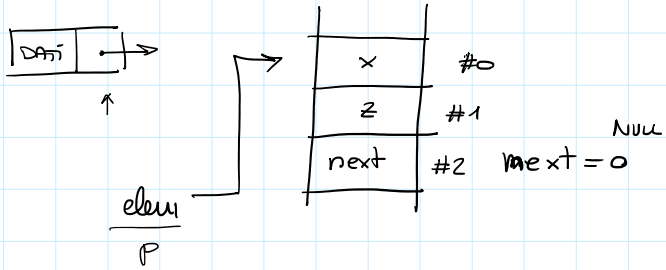
Ricerca in una lista Scorrimento di una lista (di interi) per cercare un valore $\bar{x}$

mercoledì 8 novembre 2017 08:27

```
struct { int x;
         int z; }
```



p

```
while (p != NULL && p.x ≠ x̄)
    p = p.next;
```

while (C) {B}

```
while (p != NULL) {
    IF (p.x == x̄)  { z → ; break; }
    p = p.next;
}
```

Rtarget

IF (!C) end
comp (B)
GOTO

```
①  loop:  IF=∅   Rp, fine
②            LOAD   Rp, R∅, Rx
③            IF=    Rx, Rtarget, trovato
④            LOAD   Rp, #2, Rp
⑤            GOTO   loop
   trovato:  LOAD   Rp, #1, Rz
   ...
   fine:     END
```

2 LD/ST
2 IF=
1 GOTO
5 *

$5\,T_{ch} = 5(2\tau + t_a)$

$+$

$2\,T_{exec_{LD/ST}} + 2\,T_{exec_{IF=}} + 1\,T_{exec_{GOTO}}$

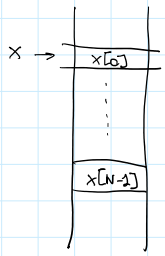
$2(2\tau + t_a) + 2(2\tau) + 1(\tau) = 9\tau + 2t_a$

Quanto costa "passare" su un elemento della lista senza trovare $\bar{x}$?

1 iterazione costo

$\underbrace{5(2\tau + t_a)}_{ch} + \underbrace{9\tau + 2t_a}_{exec} = 19\tau + 7t_a$

MAX di un vettore x (di N posizioni) (di interi positivi)



S_1 || max = x[0];

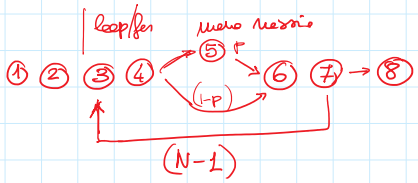
S_2 || for(int i = 1; i < N; i++)
if(x[i] > xmax) max = x[i];

max = 0;
for(int i=0; i<N; i++)
if(x[i]>max) max = x[i];

comp { S_1 ; S_2 } ⇒ comp(S_1) comp(S_2) comp(assegnamento)

comp(for( ) B) → etich : comp(B) → comp(if(e) Then)
INC Ri
IF< Ri, RN, etich
IF_VE cout
comp(then) → comp(assegnamento)
cout: ----

①	S1:	LOAD	R_{basex}, R_0, R_{max}	
②	S2:	ADD	R_0, #1, R_i	
③	etich:	LOAD	R_{basex}, R_i, R_{xi}	
④		IF<	R_{xi}, R_{max}, cout	
⑤	then:	ADD	R_{xi}, $R_\emptyset$, R_{max}	; MOV R_{xi}, R_{max}
⑥	cout:	INC	R_i	
⑦		IF<	R_i, R_N, etich	
⑧		END		



tempo di completamento

Quanto è T_c del programma

p : probabilità then
(1-p) : " else

$T_1 + T_2 + (N-1)\left(T_3 + T_4 + \left(p\,T_5 + (1-p)\cdot\emptyset\right) + T_6 + T_7\right) + T_8$

$T_1 = \underbrace{2\tau + t_a}_{ch} + \underbrace{2\tau + t_a}_{exec}$

$T_2 = 2\tau + t_a + \tau$

$(N-1)\left(\left(4+\frac{1}{2}\right)(2\tau + t_a) + (2\tau + t_a) + (2\tau) + \frac{1}{2}(\tau) + (\tau) + (2\tau)\right)$

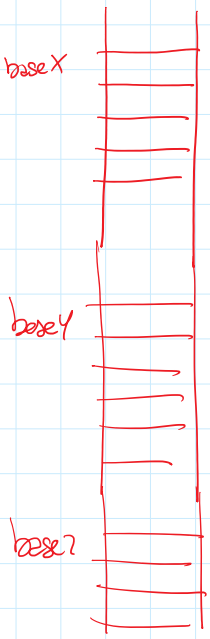
(quanto) (esecuzione: ③ ④ ⑤ ⑥ ⑦)

$T_1 + T_2 = 7\tau + 3t_a + (N-1)\left(9\tau + \frac{9}{2}t_a + \frac{15}{2}\tau + t_a\right) =$

$= 7\tau + 3t_a + (N-1)\left(\frac{33}{2}\tau + \frac{11}{2}t_a\right)$

Nuove istruzioni

mercoledì 8 novembre 2017 10:16



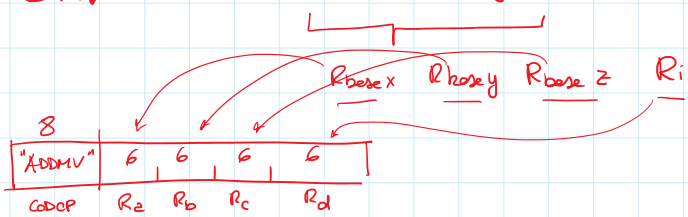
for( ) { }

z[i] = x[i] + y[i];

LOAD ... R_{xi}
LOAD ... R_{yi}
ADD R_z
STORE ... R_z

↓

ADDMV : z[i] = x[i] + y[i]



addmv0. "read" → OP, REG[IR.R_a] + REG[IR.R_d] → IND, set RDY$_{out}$M, addmv1

addmv1. (RDY$_{in}$M, OR(ESITO) = 0 –) nop, addmv1
(= 10) DATAIN → TEMP1, reset RDY$_{in}$M, REG[IR.R_b] + REG[IR.R_d] → IND
"read" → OP, set RDY$_{out}$M, addmv2
(= 11) ---

addmv2. (RDY$_{in}$M, OR(ESITO) = 0 –) nop, addmv2
(= 10) reset RDY$_{in}$M, DATAIN + TEMP1 → DATAOUT,
REG[IR.R_c] + REG[IR.R_d] → IND, "write" → OP, set RDY$_{out}$M
addmv3
(= 11) ---

addmv3. (RDY$_{in}$M, OR(ESITO), INT = 0 – –) nop, addmv3
(= 100) reset RDY$_{in}$M, IC+1 → IC, clr0
(= 101) reset RDY$_{in}$M, IC+1 → IC, halt-int
(= 11 –) ---

$$T_{addmv} = T_{ch} + T_{exec_{addmv}} = 6\tau + 4t_a$$

$T_{ch} = 2\tau + t_a$ + $T_{exec_{addmv}} = 4\tau + 3t_a$

LD
LD
ADD
ST

$4T_{ch} + 3T_{exec_{LD/ST}} + T_{exec_{ALC}}$

$4(2\tau + t_a) + 3(2\tau + t_a) +$
$\tau =$
$15\tau + 7t_a$

esempio con chiamata di funzione

mercoledì 8 novembre 2017 10:45

$\forall i \quad x[i] = f(y[i])$

$f(x) = x^2 + 1$

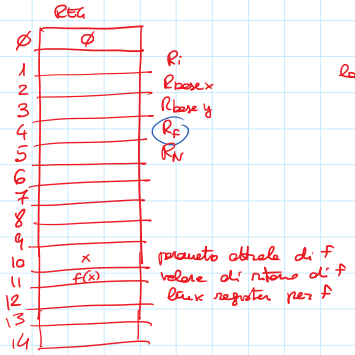
```
for(int i=0; i<N; i++)
    x[i] = f(y[i]);      CALL Rf, Rret

int f(int x) {
    return (x*x + 1);
}
```

compilazione di f:

R_{10} → x in ingresso
R_{11} → valore di ritorno
R_{12} → ritorno

```
f:  MUL  R10, R10, R11
    ADD  R11, #1, R11
    GOTO R12
```

```
    CLEAR  Ri      // ADD R0, R0, Ri
loop: LOAD   Rbasey, Ri, R10
    CALL   Rf, R12,
    STORE  Rbasex, Ri, R11
    INC    Ri
    IF<     Ri, RN, loop
    END
```



linguaggio ASM ("alto livello") → compilatore → linguaggio macchina

```
loop  CLEAR  R1
      LOAD   R3, R1, R10
      CALL   R4, R12
      STORE  R2, R1, R11
      INC    R1
      IF<    R1, R5, -4
      END
```

	6	6	6	6
CODOP				
ADD	0	0	1	
LOAD	3	1	10	
CALL	4	12		
ST	2	1	11	
ADDI	1	1	1	
IF<	1	5	-4	
END