



3) Tradurre il seguente codice in assembleur
 $F = (g+h) - (i+j)$

(0) Assegno alle variabili della memoria un indirizzo di un registro

$$F = x_{10} \quad h = x_{21} \quad j = x_{23}$$

$$g = x_{20} \quad i = x_{22}$$

$U_1 = x_{24} \quad U_2 = x_{25} \leftarrow$ var di comodo

(1) Definisco le istruzioni

$$\text{ADD } x_{24} \quad x_{20} \quad x_{21} \equiv g+h$$

$$\text{ADD } x_{25} \quad x_{22} \quad x_{23} \equiv j+i$$

$$\text{SUB } x_{10} \quad x_{24} \quad x_{25} \equiv (g+h) - (j+i)$$

2) Risolvere il problema della foto:

Istruzioni di accesso alla memoria

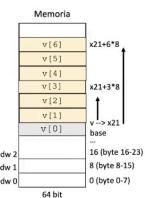
• Un esempio

```
long g, h, f;
long v[10];
...
g = h + v[3]
v[6] = g - f
```

Linguaggio C



RISC-V assembler



(0) Definisco le load dalla memoria

ld x10, x5 long g

ld x20, x6 long h

ld x21, x7 long F

ld x22, x8(x8) long U23

ld x23, h8(x8) long U26

(1) Effettuo i calcoli

sum x10, x20 x22

sub x23, x10 x21

(2) Storicozzo in memoria

sd x23, h8(x8)

(1) Risolvere il problema nella foto:

Prima un esempio semplice...

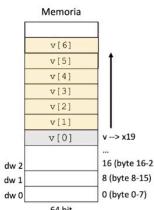
```
long i, j;
long v[10];
...
v[i] = v[j];
```

Linguaggio C



RISC-V assembler

?

(1) faccio load dei due i per $U[i]$

add x22, x20, x0 → Mi salvo il valore di "i" in x22 per non sovrascriverlo dopo.

slai x22, x22, 3

→ Preso il numero, lo moltiplico per $2^3 = 8$ byte ovvero la grandezza di long
 È come fare `syscall long`
 Ora su x22 c'è il numero effettivo di posizioni da saltare

add x23, x22, x24

→ Posiziono l'indice sull'i-esimo elemento

(2) Preparo $U[j]$

add x23, x21, x0

slai x23, x23, 3

add x23, x23, x24

ld x24, 0(x23)

] come nel punto 1 preparo l'indice $U[j]$ → Carica $U[j]$ (3) Salvo $U[i] = U[j]$

sd x22, 0(x24)

→ Salvo il valore

LS

(1) Risolvere l'esercizio in foto

Salti condizionati: costrutto if-then-else

- Attraverso le istruzioni `beq` e `bne` è possibile tradurre in assembler il costrutto `if` dei linguaggi di programmazione ad alto livello



(1) Codifico le istruzioni

`beq x20, x21, IF-THEN`

(0) Definisco registri delle variabili

$$x20 = i$$

$$x21 = j$$

`sub x22, x20, x21`

(altrimenti salta nel ramo else)

`beq x0, x0, END-IF`

Salto a END-IF per non eseguire il THEN

IF-THEN: `add x22, x20, x21` (eseguo il then)

END-IF:

(2) Risolvere l'esercizio in foto

Salti condizionati: ciclo for

- Una possibile implementazione

```

for (i=0; i<100; i++)
{
  ...
}
  
```



(1) Codifico le istruzioni

LOOP: `blt x20, x21, ENDOOP`

// resto del codice.

`addi x20, x20, 1`

`beq x0, x0, LOOP`

ENDLOOP:

(0) Definisco registri

$$x20 = i$$

$$x21 = 100$$

$$\text{addi } x21, x20, 100$$

