Architettura degli elaboratori - lezione 3

Appunti di Davide Vella 2024/2025

Claudio Schifanella
claudio.schifanella@unito.it
Link al moodle :
https://informatica.i-learn.unito.it/course/view.php?id=3106

25/02/2025

Contenuti

Associazione registri/variabili

Per passare da un linguaggio ad alto livello ad assembly, dobbiamo passare dalle variabili (come "tmp, a, b, c, somma,...") a indirizzi dei registri ("x5, x20, x21, ...").

Istruzioni aritmetiche

- Somma : add
- Sottrazione : Può succedere che si incontri la necessità di cambiare segno ad un valore. L'istruzione "sub" può essere utilizzata per questa cosa. Dando x19 = a, da a = -a, diventa (in assembly) "sub x19, x0, x19" (così diventa a = 0 - a). La sottrazione avviene grazie al complemento a 2 del secondo operando e poi avviene la somma (inv(...) per invertire un registro)
  Ogni istruzione di alto livello, può essere riscritta come più istruzioni di basso livello, ad esempio :

```
f = a + b - c
```

Diventa :

```
add x19, x20, x21
sub x19, x19, x22
```

Quindi facciamo prima la somma tra a e b (rispettivamente x20, x21) e messo in f (x19) e poi facciamo f = f - c (c = x22)

Esercizio :

```
f = (g + h) - (i + j)
```

Var	Reg
f	x20
g	x21
h	x22
i	x23
j	x24

```
add x21, x21, x22
add x23, x23, x24
sub x20, x21, x23
```

```
g = g + h;
i = i + j;
f = g - i;
```

Accesso alla memoria

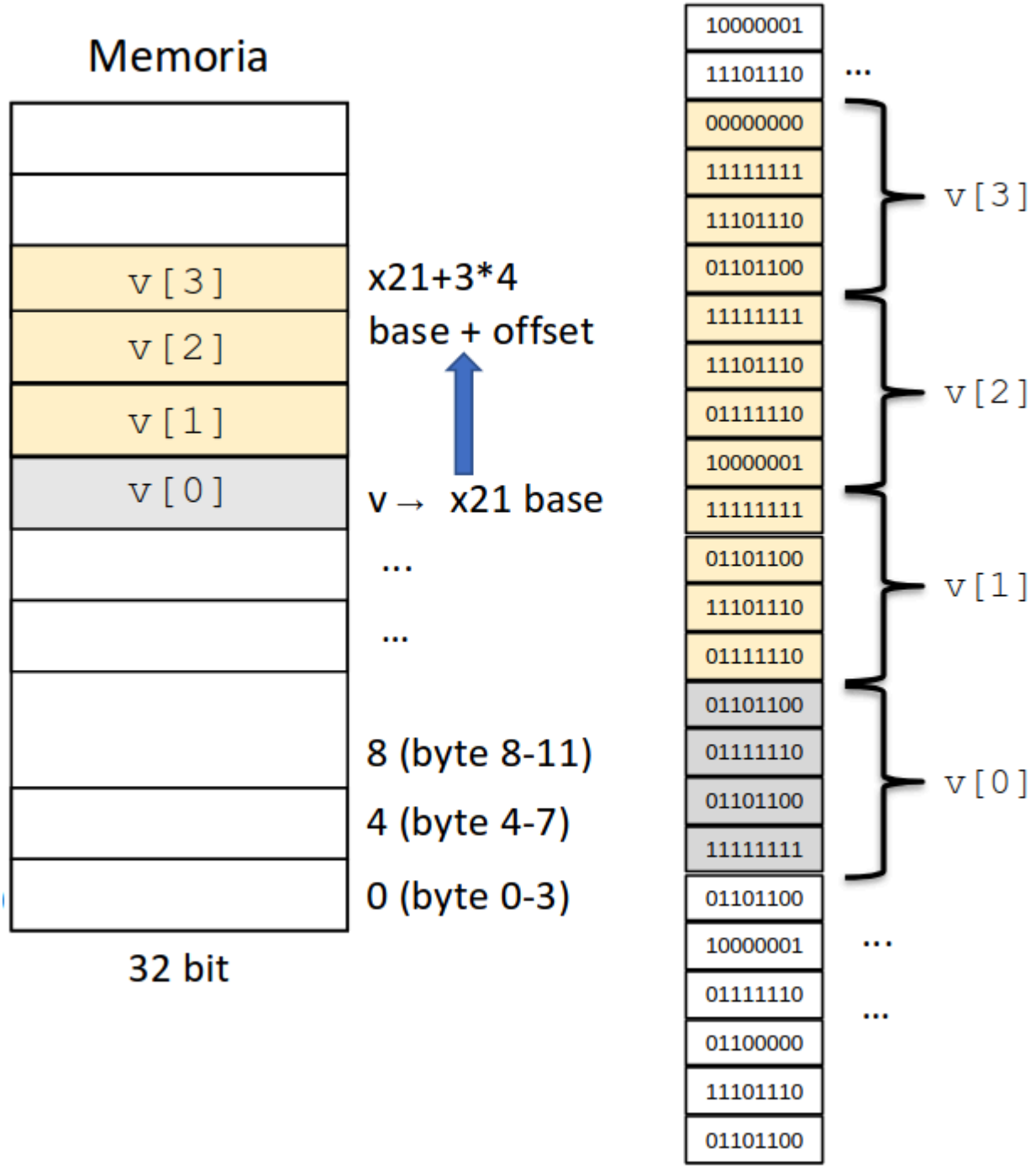
La memoria può essere vista come un grande array di celle (da 1 byte l'una). Prendiamo come esempio il seguente schema :

Valori	Cella
	...

Valori	Cella
	...
	5
	4
	3
00111010	2
10111011	1
11111110	0

Istruzione Load

L'istruzione load permette di copiare un dato dalla memoria ad un registro. Prendiamo il seguente esempio :



Se cerchiamo di fare

```
a = v[3];
```

Dobbiamo usare l'istruzione **lw** (load word) nel seguente modo :
lw (destinazione della lettura), offset (indirizzo di base)
Un esempio è :

Var	registri
a	x5
v (array int)	x21

```
lw x5, 12(x21)
```

L'offset di 12 ci serve perché dobbiamo andare al primo indirizzo di dove è salvato il valore v(3). Prendiamo lo spazio occupato da un int (4 byte) e moltiplichiamo per l'indirizzo a cui vogliamo arrivare (3), quindi abbiamo 12. **NOTA**, x21 in questo caso è un registro che contiene un indirizzo.
L'offset può valere al massimo -2048 o +2047.

little endian

L'indirizzo della parola identifica il byte meno significativo

Istruzione store

Per scrivere in memoria usiamo l'istruzione **sw** (store word). La notazione è :
sw (dato da scrivere in memoria), offset(indirizzo di base).

Istruzione sh

Come sw, sh serve per scrivere, ma a differenza di sw (che scrive 4 byte (1 word)), con sh scriviamo una halfword (2 byte), ovvero i meno significativi, sintassi :
sh (dato da scrivere in memoria), offset(indirizzo di base).

esericizio scrittura/lettura

```
g = h + v[3]
v[6] = g - f
```

Var	Reg
g	x5
h	x9
f	x19
v (array)	x21

```
lw x2, 12(x21)
add x5, x9, x2
sub x2, x5, x19
sw x2, 24(x21)
```

In questo caso, usiamo x2 come registro d'appoggio.

Domande poste durante la lezione (forse utili per esame)

- Perché usiamo i registri associate alle variabili (a = x5, b = x20...): Usiamo i registri associati alle variabili perché la ALU ha come input solo il contenuto dei registri (in RISC-V).
- Tra cosa si fa la somma? : Somma i valori contenuti nei registri, qualsiasi cosa essi abbiano (valori, puntatori).
- Quando usiamo i registri? : Se non siamo in un contesto particolare, possiamo usare tutti e 31 (x0 = 0 sempre).
- Con l'istruzione Load (lw), perché dobbiamo usare l'offset? : Perchè altrimenti (se guardiamo l'esempio sopra), dobbiamo aggiornare ogni volta il valore di x21, es :

  ```
  a = v[0];
  b = v[1];
  c = v[2];
  ```

  In assembly sarebbe (senza offset) :

  ```
  lw x5, x21
  addi x21, x21, 4
  lw x6, x21
  addi x21, x21, 4
  lw x7, x21
  ```

  Con offset diventa :

  ```
  lw x5, 0(x21)
  lw x5, 4(x21)
  lw x5, 8(x21)
  ```

- Perché con sh scriviamo prima i bit meno significativi e poi quelli più significativi? : perché in un intero, occupiamo prima i bit iniziali, poi quelli dopo.