Università degli Studi di Milano «La Statale»
Dipartimento di Informatica

Esempi di programmi in MIPS

# Variabili e RAM
## 2/2

Marco Tarini

67

## Vettori (array) Esempio

```
.data
voti: .word 28 21 30 27 24

.text
la $6 voti  # $6 = l'indirizzo dell'array voti
lw $28 ($6) # copio voti[0] (cioè 28) in $28
lw $28 12($6) # copio voti[3] (cioè 27) in $28
```

68

## Stesso esempio, ma con un vettore di half. Cosa cambia?

```
.data
voti: .half 28 21 30 27 24

.text
la $6 voti  # $6 = l'indirizzo dell'array voti
lh $28 ($6) # copio voti[0] (cioè 28) in $28
lh $28 6($6) # copio voti[3] (cioè 27) in $28
```

Load half

3x**2**, perché la dimensione di un half è di 2 byte

69

## Stesso esempio, ma con un vettore di byte. Cosa cambia?

```
.data
voti: .byte 28 21 30 27 24

.text
la $6 voti  # $6 = l'indirizzo dell'array voti
lb $28 ($6) # copio voti[0] (cioè 28) in $28
lb $28 3($6) # copio voti[3] (cioè 27) in $28
```

load byte

3x**2**, perché la dimensione di un half è di 2 byte

70

## Esempio: volgiamo in maiuscolo un carattere di una stringa (che è un vettore di byte)

```
.data
saluto: .asciiz "ciaomamma"

.text
la $6 saluto # $6 = l'indirizzo dell'array
lb $5 4($6)  # copio saluto[4] (cioè 'm') in $5
add $5 $5 -32 # volgo la lettera in maiuscolo
sb $5 4($6)  # copio $5 in saluto[4] in
```

store byte

4x**1**, perché la stringa è un array di byte

Sottrarre 32 è un modo di volgere in maiuscolo una lettera minuscola (vedi tabella ASCII)

71

## Struct (o Object, o record)

- Nei linguaggi **ad alto livello**, è comodo organizzare i dati incapsulandoli come «campi» di uno stesso «oggetto»
  - Ogni campo descrive un aspetto dell'oggetto
  - Nota: a differenza degli elementi di un array (o vettore), ogni campo di un oggetto può essere di tipo diverso
- Chiamato con vari nomi e sintassi, come:
  - **Struct** (in C, Go, C#, Ruby, Haskell, Swift, Rust…)
  - **Data Class** (Kotlin)
  - **Class** (Java, C++, Python, Ruby, Kotlin)
  - **Record** (Pascal, Ada)
  - **{…}** (cioè un literal di tipo «object») (in JavaScript, JSON, F#, ML/OCaml)
  - **Interface** (Typescript)

72

## Struct (o Object, o record): esempio

In Go:

```
type Dog struct {
    name string
    isAGoodBoy bool
    age int
}

fufo : Dog

fufo.name = "Fufo"
fufo.isAGoodBoy = true
fufo.age = 8
```

In C:

```
struct Dog{
   char name[5];
   bool isAGoodBoy;
   int age;
};

struct Dog fufo;

fufo.name = "Fufo";
fufo.isAGoodBoy = true;
fufo.age = 8;
```

Esempio di uso: buon compleanno, Fufo!

```
fufo.age = fufo.age + 1
```

```
fufo.age++;
```

73

## Struct (o Object, o record): esempio B

In Go:

```
type Dog struct {
    name string
    isAGoodBoy bool
    age int
}

fufo := Dog{
    name:       "Fufo",
    isAGoodBoy: true,
    age:        8,
}
```

In C:

```
struct Dog{
   char name[5];
   bool isAGoodBoy;
   int age;
};

struct Dog fufo =
{
  "Fufo", true, 8
}
```

Esempio di uso: buon compleanno, Fufo!

```
fufo.age = fufo.age + 1
```

```
fufo.age++;
```

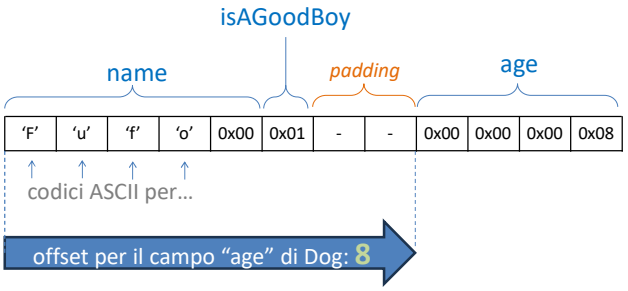
74

# Come lo traduce il compilatore?

In MIPS-32:

```
.data

fufo: .asciiz "Fufo"
      .byte 1
      .word 8
```




Esempio di uso: buon compleanno, Fufo!

```
.text

la $10 fufo
lw $2 8($10)
addi $2 $2 1
sw $2 8($10)
```

75

# Note ai lucidi precedenti

- Per semplicità, l'esempio di compilazione segue la versione del C, un linguaggio un po' più a basso livello del Go
  - In Go, un oggetto di tipo "string" è a sua volta una struttura che contiene la locazione di memoria a cui stanno i caratteri (in UTF-8), e la lunghezza di questo vettore
  - Nella versione C, la campo nome è direttamente un vettore di caratteri, (in ASCII) terminato dal carattere 0 (come in MIPS)…
  - …e la sua lunghezza è fissa (5 byte, compreso il terminatore) I cani, qui, hanno nomi di AL MASSIMO 4 lettere!
- L'assembler determina come posizionare i vari campi nello «struct»…
  - Qui, ad es, aggiunge un «padding» di 2 byte per allineare il campo «age» (che è un word e deve essere allineato a 4 byte)
  - Non è stato necessario invece un padding per il bool, per il quale basta usare un byte (che non richiede allineamento)
- …e (nel segmento .text del programma) accede ai vari campi di conseguenza
  - Ad esempio, accede al campo «age» (in lettura prima, in scrittura poi) applicando l'offset di 8, perché «age» compare un totale di 8 byte dopo l'inizio dello struct (5 di nome, 1 del boolean, e 2 di padding)

76

## Direttive Assembler per il segmento dati

`.word` : «i valori che seguono vanno memorizzati in un word ciascuno (4 byte)»

`.half` : «i valori che seguono vanno memorizzati in un half-word ciascuno (2 byte)»

`.byte` : «i valori che seguono vanno memorizzati in un singolo byte ciascuno»

`.space N` : «lascia N byte non utilizzati prima del dato successivo»
(ad esempio, il programma scriverà in questo spazio prima di leggervi)

77

## Allineamento dati: ripasso

- L'accesso a memoria allineato su *n* byte se ogni dato di dimensione *n* byte comincia ad un indirizzo multiplo di *n*
  - con *n* potenza di 2
- In MIPS l'accesso a word è allineato a 4:
  - il loro indirizzo deve essere multiplo di 4
  - altrimenti: viene generato un errore a runtime (una «trap»)
- In MIPS l'accesso agli half-word è allineato a 2:
  - il loro indirizzo deve essere multiplo di 2
  - altrimenti: viene generato un errore a runtime (una «trap»)

78

## Direttive Assembler per il segmento dati

`.align n` :

«Lascia qui un certo numero di byte vuoti (0 o più), detto «padding», prima del prossimo dato, in modo da rendere l'indirizzo in memoria divisibile per $2^n$ » dove `n` vale 1 o 2

- per allineare i world (4 byte) usare `n` =2
- per allineare gli half (2 byte) usare `n` =1
- Nota:
  la direttiva ".word" include un .align 2 e
  la direttiva ".half" include un .align 1

79

## Esercizi

(da provare su http://mipsweb.di.unimi.it )

- Nell'esempio sopra, scrivi un brano di codice assembly che «flippa» il valore isAGoodBoy di Fufo
  - quindi, lo fa diventare cattivo se era buono, e viceversa
- Scrivi un programma assembly che ha, come dati, un array con i primi 6 numeri primi (espressi come un half), e, come istruzioni, un codice che (attraverso una sequenza di addizioni consecutive), memorizza in un registro la loro somma totale.
- Genera e testa due esempi di codice assembly a piacere che generino un errore, rispettivamente…
  - di overflow
  - di accesso ad un word non allineato

  (osserva l'errore riportato).

80