Il linguaggio assembly

- Direttive
- Chiamate di sistema (system call)
- Esempi

1

Direttive

- Le direttive (data layout directives) danno delle indicazioni all'assemblatore sul contenuto di un file (istruzioni, strutture dati, ecc.)
- Le direttive iniziano tutte con il carattere "." (punto)

2

Un programma completo

```
# Somma i valori di un array
        .data
array:  .word 1,2,3,4,5,6,7,8,9,10 # dichi.ione array
        .text
        .globl main
main:
        li $s0,10           # $s0 <- numero elementi
        la $s1,array        # $s1 <- base_address
        li  $s2,0           # $s2 contatore cicli
        li $t2,0            # azzera $t2 accumulatore
loop:   lw $t1,0($s1)       # $t1 <- mem[$s1]
        add $t2,$t1,$t2     # $t2 <- $t2 + $t1
        addi $s1,$s1,4      # $s1 <- indir.success.vo
        addi $s2,$s2,1      # contatore cicli ++
        bne  $s2,$s0,loop   # test terminazione
```

3

Traduzione in Java

```
class ProvaWhile {
  static int[] array = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

 public static void main( String[] args ) throws
  Exception {
  int i, somma, dim;
  dim = 10; //in $s0
  i = 0;    //in $s2
  somma = 0;      //in $t2
  do {
      somma = array[i] + somma;  i=i+1;
  } while (i != dim);
  //System.out.println(" Somma = "+somma);
 }
} // end classe
```

4

Direttive

- `.data`

  gli elementi successivi sono memorizzati nel segmento dati
- `.asciiz "str"`
  - Memorizza la stringa "str" terminandola con il carattere null
  - .ascii "str" ha lo stesso effetto, ma non aggiunge alla fine il carattere null
- `.byte v1,…,vn`
  - Memorizza gli n valori v1, . . . ,vn in byte consecutivi di memoria

5

Direttive (cont.)

- `.word w1, . .,wn`
  - Memorizza gli n valori su 32-bit w1, .., wn in parole consecutive di memoria.
- `.half h1, . .,hn`
  - Memorizza gli n valori su 16-bit h1, .., hn in halfword (mezze parole) consecutive di memoria
- `.space n`
  - Alloca uno spazio pari ad n byte nel segmento dati
- `.text`
  - Memorizza gli elementi successivi nel segmento testo (questi elementi possono essere solo istruzioni o parole)

6

Direttive (cont.)

- `.globl lab`
  - Dichiara lab come etichetta globale (ad essa è possibile fare riferimento da altri file)
- `.align n`
  - Allinea il dato successivo a blocchi di 2^n byte: ad esempio

    .align 2 = .word allinea alla parola il valore successivo

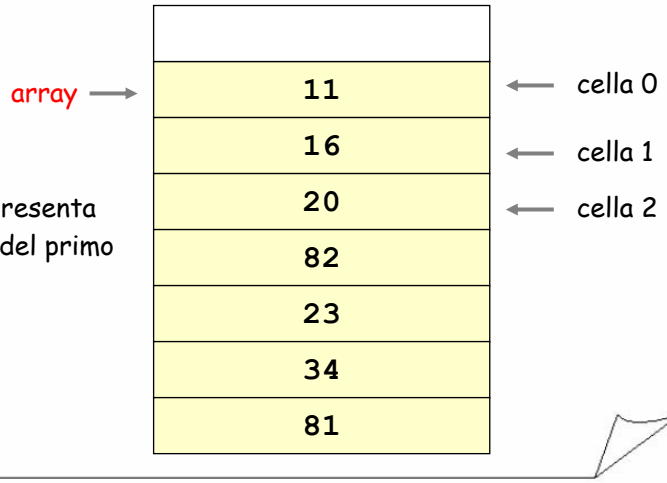
    .align 1 = .half allinea alla mezza parola il valore successivo

    .align 0 elimina l'allineamento automatico delle direttive .half, .word, .float e .double fino a quando compare la successiva direttiva .data

7

Direttive .word - esempio

```
array: .word 11, 16, 20, 82, 23, 34, 81
```



- array rappresenta l'indirizzo del primo elemento

8

Direttive: esempio

```
# Somma numeri memorizzati nell'array "vet"
# $t0 -> indice dell'array
# $t1 -> accumulatore
# $t3 -> offset ultimo elemento
    .data
vet:.word 10, 20, 30, 40, 50, 60
    .text
main:
     move $t0, $zero      # inizializ. ind. dell'array
     addi $t3, $zero, 20  # offset ultimo elemento
     lw   $t1, vet($t0)   # carica 1^ valore in $t1
loop:
     addi $t0, $t0, 4     # avanza indice dell'array
     lw   $t2, vet($t0)   # $t2 <- vet[$t0]
     add  $t1, $t1,$t2    # $t1 <- $t1 + $t2
     bne  $t0, $t3, loop # se $t0 < $t3 vai a "loop"
```

9

System call

- Il kernel MIPS fornisce alcuni semplici servizi mediante chiamate di sistema (system call) predefinite
- Ogni system call è costituita da:
  - un codice (dell'operazione)
  - degli argomenti (opzionali)
  - dei valori di ritorno (opzionali)

10

System call per stampare

- print_int: stampa sulla console un numero intero
- print_float: stampa sulla console il numero in virgola mobile con singola precisione
- print_double: stampa sulla console il numero in virgola mobile con doppia precisione
- print_string: stampa sulla console una stringa

11

System call per leggere

- read_int: legge un intero
- read_float: legge un float
- read_double: legge un double
- read_string: legge una stringa di caratteri di lunghezza `$a1` scrivendola in un buffer (`$a0`) e terminando la stringa con il carattere null

12

System call per memoria/terminazione

- sbrk: restituisce il puntatore (indirizzo) ad un blocco di memoria acquisito dinamicamente
- exit: interrompe l'esecuzione di un programma

13

Codici delle system call (cont.)

Servizio	Codice	Argomenti	Risultato
print_int	1	$a0 = integer	
print_float	2	$f12 = float	
print_double	3	$f12 = double	
print_string	4	$a0 = string	
read_int	5		integer (in $v0)
read _float	6		float (in $f0)
read _double	7	$f12 = double	double (in $f0)
read _string	8	$a0 = buf; $a1 =lung	
sbrk	9	$a0 = quantità	indirizzo (in $v0)
exit	10		

14

Come fare una system call

- Per richiedere un servizio al sistema con una (syscall) :
  1. Inserire il codice della chiamata nel registro $v0
  2. Inserire gli argomenti nei registri $a0 - $a3 ($f12 - $f15)
  3. Eseguire l'istruzione syscall
  4. Il valore di ritorno (se c'è) è nel registro $v0 ($f0)

15

System call - Esempio 1

```
#Programma che stampa: "La risposta è 5"
      .data
str:  .asciiz "La risposta è"
      .text
      li $v0, 4    # $v0 <- codice di print_string
      la $a0, str  # $a0 <- indirizzo della stringa
      syscall      # stampa della stringa

      li $v0, 1    # $v0 <- codice print_integer
      li $a0, 5    # $a0 <- intero da stampare
      syscall      # stampa dell'intero

      li $v0, 10   # $v0 <- codice della exit
      syscall      # esce dal programma
```

16

System call - Esempio 2

```
# Il programma stampa "Dammi un intero: "
# e legge l'intero immesso
        .data
prom:   .asciiz "Dammi un intero: "
        .text
main:
        li $v0, 4       # $v0 <- codice print_string
        la $a0, prom    # $a0 <- indirizzo stringa
        syscall         # stampa la stringa

        li $v0, 5    # $v0 <- codice della read_int
        syscall      # legge un intero in $v0

        li $v0, 10      # $v0 <- codice della exit
        syscall         # esce dal programma
```

17

System call - Esempio 3

```
# Programma che calcola il fattoriale (iterativo)
        .data
prompt: .asciiz "Inserisci un numero intero "
output: .ascii "Il fattoriale è:"
        .text

main:   li $v0, 4       # $v0 <- codice print_string
        la $a0, prompt # $a0 <- indirizzo stringa
        syscall         # stampa la stringa
        li $v0, 5 # $v0 <- codice della read_int
        syscall   # legge l'intero in $v0
# continua ....
```

18

System call - Esempio 3 (cont.)

```
# calcola il fattoriale
        li $t0, 1 # inizializzo $t0 contatore cicli
        li $t1, 1 # inizializzo $t1 accumulatore
                  # che conterrà il risultato n!
loop:   mul $t1, $t1, $t0  # $t1 <- $t1 * $t0
        addi $t0, $t0, 1   # incr. $t0 cont cicli
        ble $t0, $v0, loop # se $t0 ≤ $v0 goto loop

# continua ....
```

19

System call - Esempio 3 (cont.)

```
# stampa il risultato
    li $v0, 4       # $v0 <- codice print_string
    la $a0, output  # $a0 <- indirizzo stringa
    syscall         # stampa la stringa

    li $v0, 1       # $v0 <- codice della print_int
    move $a0, $t1   # $a0 <- n!
    syscall         # stampa n!

    li $v0, 10      # $v0 <- codice della exit
    syscall         # esce dal programma
   # fine programma
```

20