Corso di Laurea in Informatica
Architettura degli elaboratori
a.a. 2020-2021

Programmazione Assembly: procedure (1)

Passaggio parametri attraverso registri
Syscall

Claudia Raibulet
claudia.raibulet@unimib.it

Sistema Operativo e Syscall

- **In un sistema reale esiste il Sistema Operativo...**
- **...che è un insieme di programmi che:**
  - stanno in un'area (protetta) di memoria
  - svolgono funzioni di utilità generale (in particolare, I/O) richiamabili dai programmi utente
- **La struttura generale e le funzioni di un SO saranno trattate nell'insegnamento di Reti e Sistemi Operativi (II anno)**
- **I meccanismi base di chiamata al SO sono trattati nel seguito di questo insegnamento**
- **Il simulatore MIPS fornisce alcune funzioni elementari che simulano alcune funzionalità base del SO...**
- **...richiamabili attraverso il meccanismo di syscall, concettualmente analogo a una chiamata a procedura**

Syscall

- **Analogo a una chiamata a procedura**
- **Convenzioni per le syscall: Tabella a pag. A43 (Appendice A)**
- **Impostare i parametri nei registri $a0-$a3 (come da tabella)**
- **Impostare nel registro $v0 il codice della chiamata**

- **Syscall**
- **Syscall essenziali:**
  - exit2 codice 17: terminazione del programma!
  - read_int
  - print_int codice 1 (parametro passato per valore!!)
  - read_string (guardare e capire bene i parametri!!!)
  - print_string (parametro passato per indirizzo!!)
  - ....

Syscall

Service	System call code	Arguments	Result
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		double (in $f0)
read_string	8	$a0 = buffer, $a1 = length	
sbrk	9	$a0 = amount	address (in $v0)
exit	10		
print_char	11	$a0 = char	
read_char	12		char (in $a0)
open	13	$a0 = filename (string), $a1 = flags, $a2 = mode	file descriptor (in $a0)
read	14	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars read (in $a0)
write	15	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars written (in $a0)
close	16	$a0 = file descriptor	
exit2	17	$a0 = result	

FIGURE A.9.1 System services.

Syscall (esempio)

```
# il codice seguente stampa la stringa «La risposta è 5 »:
.data
str: .asciiz “La risposta è “
numero: .word 5
.text
.globl main
main:
li $v0, 4 # Codice della chiamata di sistema per print_str
la $a0, str # Indirizzo della stringa da stampare (passato per indirizzo!!!)
syscall # Stampa la stringa

li $v0, 1 # Codice della chiamata di sistema per print_int
lw $a0, numero # Numero intero da stampare (passato per valore!!!)
syscall # Stampa l’intero
```

Esercizio

- **Si chiede di calcolare la lunghezza di una stringa memorizzata partendo dall'indirizzo (i.e., etichetta) "stringa". La stringa finisce con il carattere '\0'. Si chiede di memorizzare la dimensione della stringa in memoria dopo la stringa in un word.**
- **Si chiede di scrivere la stringa e la sua dimensione sullo schermo tramite le syscall.**

Esercizio – Soluzione senza procedure

```
        .data
stringa:    .asciiz "Hello World!"
#13 byte per memorizzare la stringa
#12 byte lunghezza della stringa
dim:        .word 0
charfine:   .asciiz ""


        .text
        .globl main
main:
        la $t0, stringa
        add $t2, $zero, $zero
        #in $t2 sara' calcolata la
        #dimensione della stringa
        la $s5, charfine
        lb $s1, 0($s5)
        #lb – load byte – legge un
        #carattere rappresentato
        #su 1 byte
ciclo:
        lb $s7, 0($t0) #legge un carattere
                       #dalla stringa
        beq $s7, $s1, fine #verifica se e' il
                           #carattere \0 di fine stringa
        addi $t2, $t2, 1 #incrementa il contattore dei
                         #caratteri letti dalla stringa
        addi $t0, $t0, 1 #incrementa l'indirizzo da
                         #dove leggere il prossimo
                         #carattere nella stringa
        j ciclo
fine:
        la $t3, dim
        sw $t2, 0($t3) #salva la dimensione calcolata
                       #in memoria
        # scrive la stringa sullo schermo
        li $v0, 4
        la $a0, stringa
        syscall

        # scrive la dim della stringa sullo schermo
        #leggendo la dimensione dalla memoria
        li $v0, 1
        la $t0, dim
        lw $a0, 0($t0)
        syscall
```

Esercizio – Soluzione con procedure

```
.data
stringa:  .asciiz "Hello World!"
dim:      .word 0

          .text
          .globl main
main:
          la $a0, stringa
          jal calcola_dim_stringa
          # la dimensione della stringa
          # e' in $v0 al ritorno della procedura
          # salva la dim in memoria
          la $t7, dim
          sw $v0, 0($t7)

          # scrive la stringa sullo schermo
          li $v0, 4
          la $a0, stringa
          syscall

          # scrive la dim della
          # stringa sullo schermo
          li $v0, 1
          la $t0, dim
          lw $a0, 0($t0)
          syscall
```

```
          .data
charfine: .asciiz ""
          .text
.globl calcola_dim_stringa

calcola_dim_stringa:
          add $t2, $zero, $zero
          la $t5, charfine
          lb $t1, 0($t5)

ciclo:    lb $t7, 0($a0)
          beq $t7, $t1, fine
          addi $t2, $t2, 1  # incrementa la dim
          addi $a0, $a0, 1 # incrementa l'indirizzo
          # per il prossimo elemento della stringa
          j ciclo
          move $v0, $t2
          jr $ra
```